COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY

CHENNAI 600025

EC5612 WIRELESS COMMUNICATION AND NETWORKING LABORATORY –SEMESTER VI
(R-2019)

RECORD

SUBMITTED BY:

2021105

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERNG

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

ANNA UNIVERSITY

COLLEGE OF ENGINEERING GUINDY

BONAFIDE CERTIFICATE

NAME:

DEPT: B.E. ECE

REGISTER NUMBER: 2021105

It is certified that this is the bonafide record of the work done by the above-mentioned student in the **WIRELESS COMMUNICATION AND NETWORKING LAB-EC5612** (2019-reg) during the period January 2024 - May 2024.

Signature of Lab-In-Charge

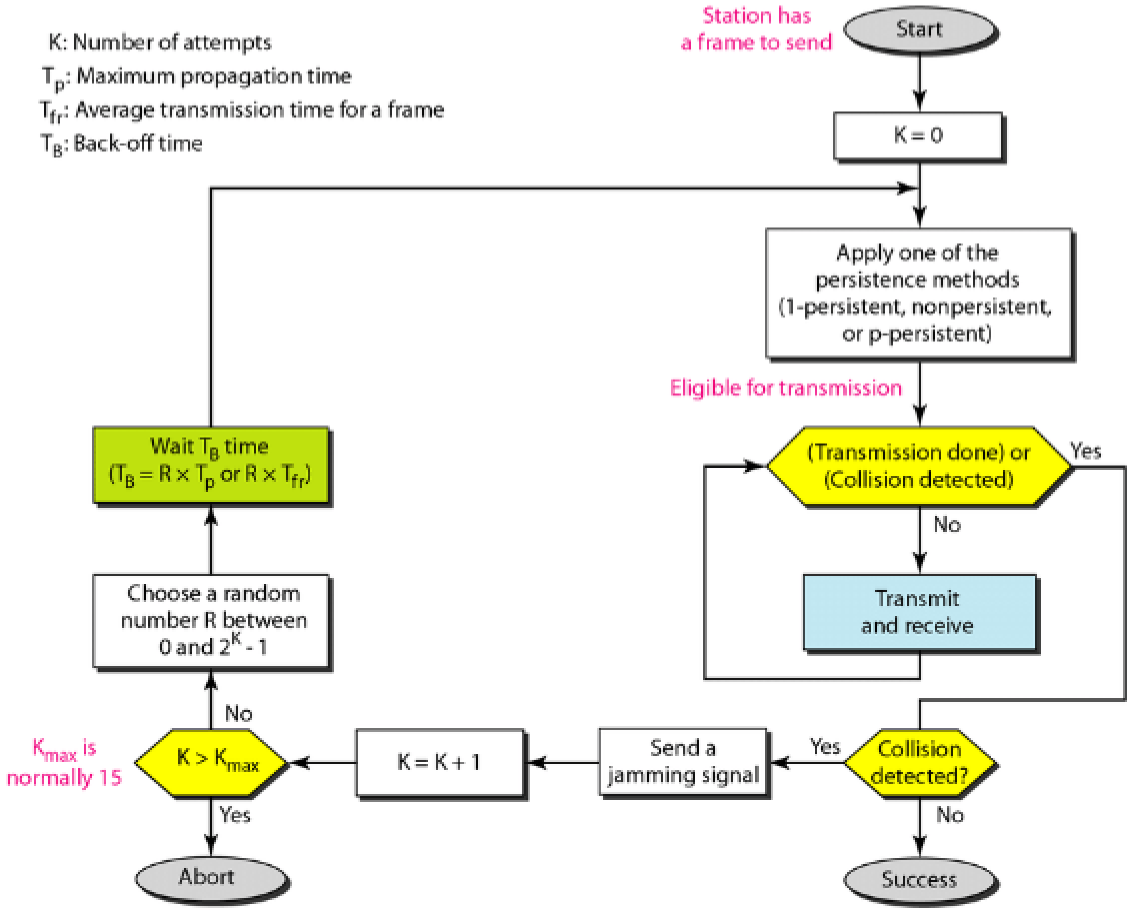
Signature of theHead of the Department

Submitted for the practical exam held on ________________

Signature of the Internal Examiner

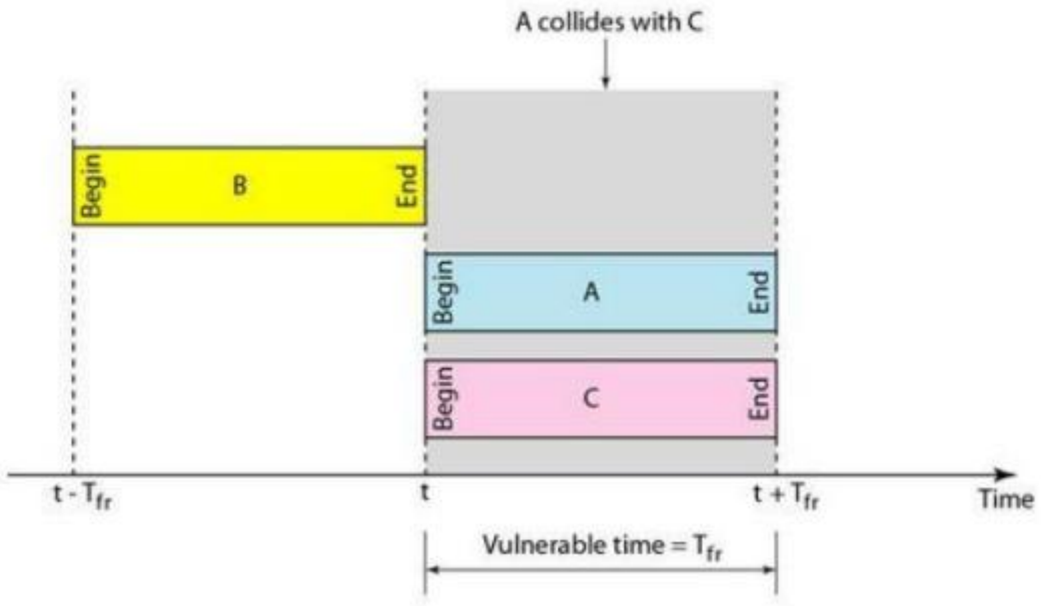
INDEX

CSMA/CA:



K: Number of attempts
T_p: Maximum propagation time
T_{fr}: Average transmission time for a frame
T_B: Back-off time
Station has a frame to send
Start
K = 0
Apply one of the persistence methods (1-persistent, nonpersistent, or p-persistent)
Eligible for transmission
(Transmission done) or (Collision detected)
Yes
No
Transmit and receive
Collision detected?
Yes
No
Success
Send a jamming signal
K = K + 1
$K > K_{max}$
K_{max} is normally 15
Yes
Abort
No
Choose a random number R between 0 and $2^K - 1$
Wait T_B time ($T_B = R \times T_p$ or $R \times T_{fr}$)

SLOTTED ALOHA



EXP.NO:1 **DATE:**	**PERFORMANCE STUDIES OF RANDOM MAC PROTOCOLS**

<u>AIM:</u>

To study and analyze the following MAC protocol.

1)Slotted aloha

2)CSMA/CD

3)CSMA/CA

<u>SOFTWARE REQUIRED:</u>

NETSIM SOFTWARE

<u>THEORY:</u>

SLOTTED ALOHA:

This allows station to transmit data only at specific time slots. So, station cannot

transmit whenever it wants. In this, the vulnerable time is reduced by half. The throughput of the slotted ALOHA is S = Ge-G which is maximum when G = 1 (37%).

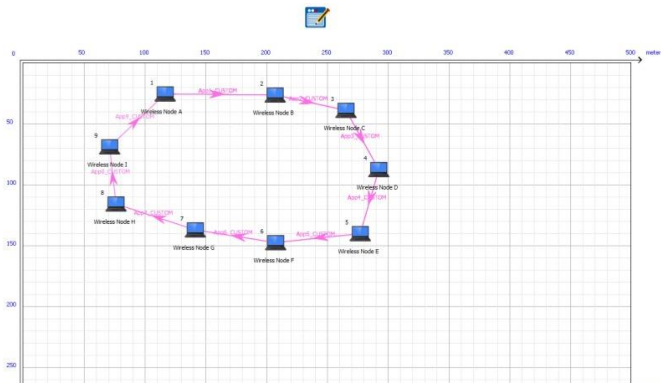
CSMA/CD:

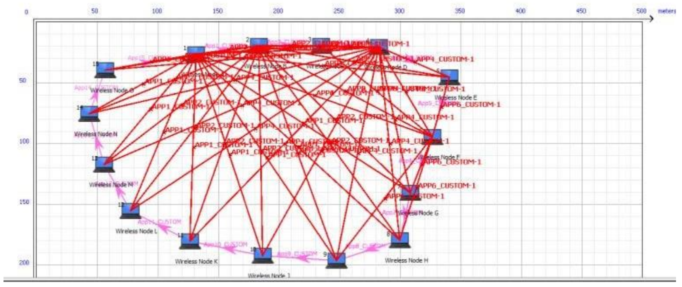
Carrier Sense Multiple Access (CSMA) with collision detection (CD) is a Medium

Access Control Method, used notably in Ethernet Technology. CSMA/CD is used to

improve the performance of pure CSMA by terminating transition as soon as a collision is detected. Thus, shortening the time required before a retry can be attempted

CONNECTING DIAGRAM:





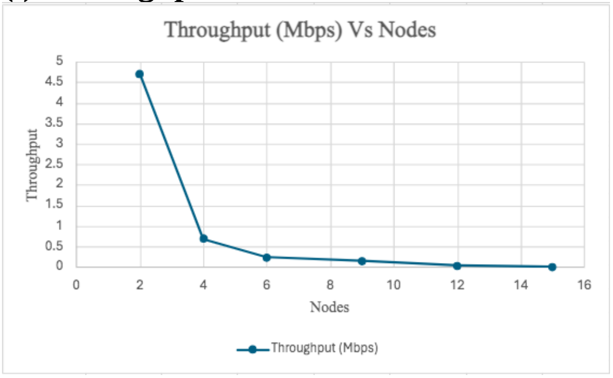
TABULATION:

1.SLOTTED ALOHA:

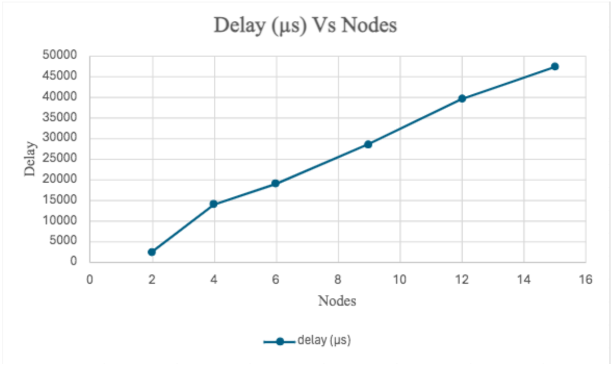
NODES	THROUGHPUT(Mbps)	DELAY(µs)
2	4.697446	2459.714214
4	0.682419	14060.91614
6	0.232576	19093.42859
9	0.142882555	28539.77959
12	0.025514583	39564.51125
15	0.008714133	47376.666667

PLOTTED GRAPHS:

(i) Throughput Vs Nodes



(ii) Delay Vs Nodes



PROCEDURE:

SLOTTED ALOHA

1. Select new → legacy network → slotted aloha

2. Continue the same step as mentioned for pure aloha

3. Calculate G and S

4. Plot:

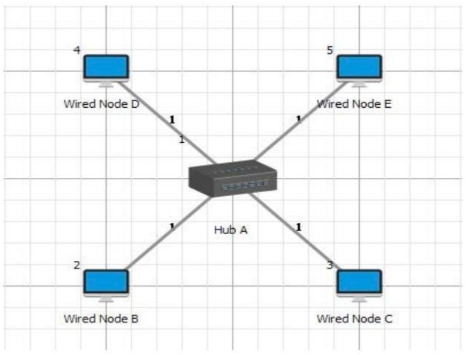
• Number of packets transmitted vs number of node

• Throughput vs nodes and delay vs nodes

5. Bandwidth is and packet time = and packet size

6. Calculate number of packets per packet time and plot it against the calculated value of throughput per packet time

CONNECTING DIAGRAM:



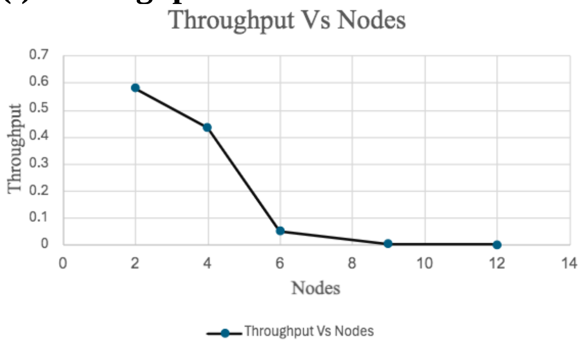
TABULATION:

1. CSMA/CA:

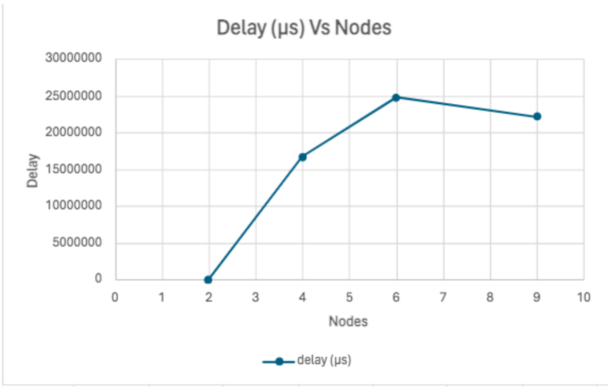
NODES	THROUGHPUT(Mbps)	DELAY(µs)
2	0.579328	1211.2
4	0.432452	16687529.91
6	0.050224	24770461.67
9	0.003504	22182629.69
12	0.000194667	7472

PLOTTED GRAPHS:

(i) Throughput Vs Nodes



(ii) Delay Vs Nodes



<u>CSMA/CD</u>

1. In NETSIM, select new advanced wireless network manet

2. Create / design a network with two wireless nodes

3. Configure the network

4. Click and drop two wireless nodes onto the simulated network

5. Position of node 1 as (50,100) and node 2 as (100,150)

6. Disable Tcp on all wireless nodes

    - Pathloss model: log distance
    - Pathloss exponent (n): 2
    - Model traffic in the network

7. Application property

    - Type: custom
    - Source id: 1
    - Destination id: 2
    - Packet size distribution: constant
    - Packet size value: 1460

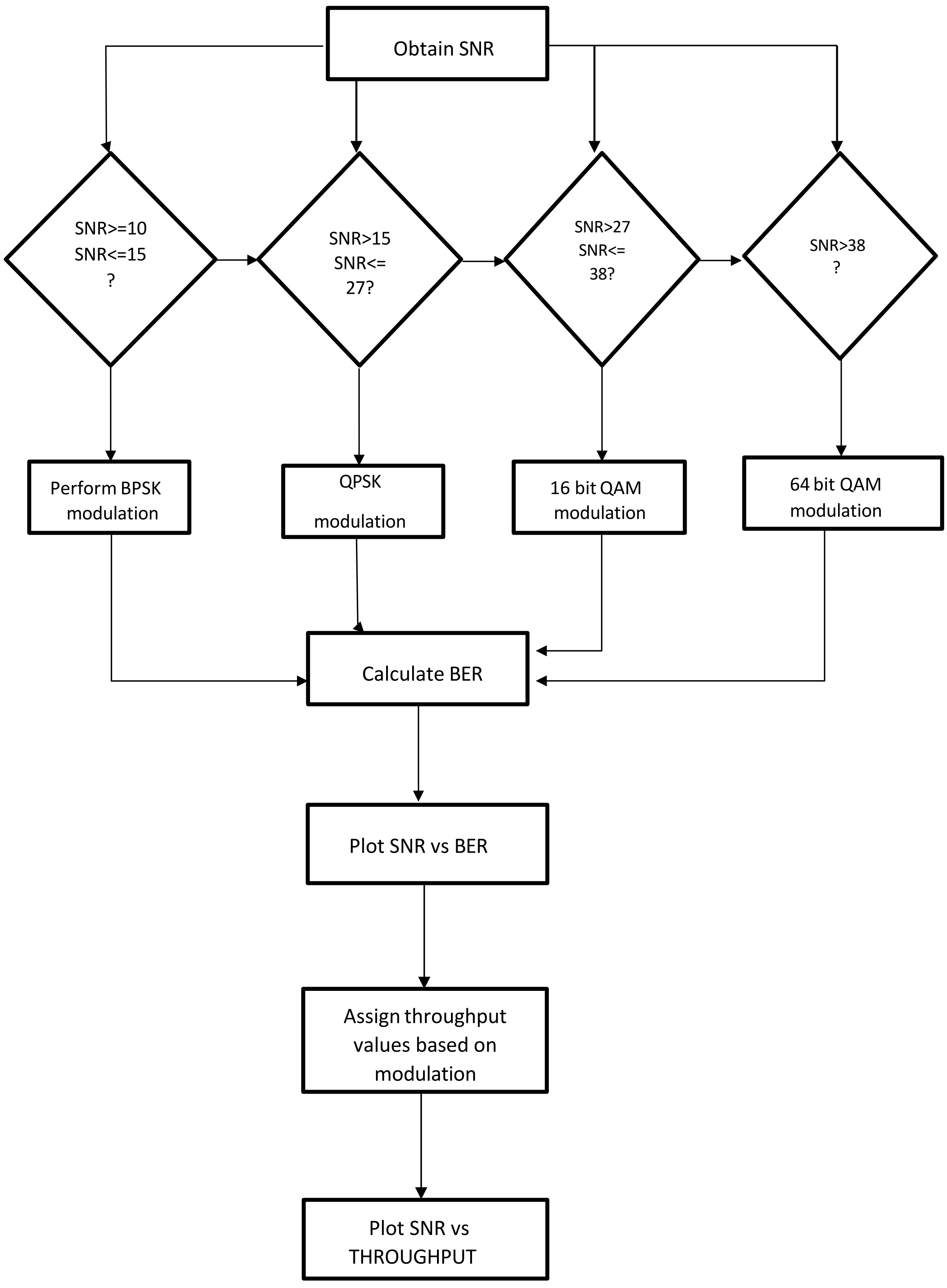
8. Set simulation time for 10 s and run the experiment

9. Add more nodes and repeat the same process and compare throughput and delay for different nodes

<u>RESULT:</u>

Thus, slotted aloha, CSMA/CD protocols of MAC layers are analyzed using NETSIM and throughput for all protocols are calculated and computed.

FLOWCHART:


Obtain SNR
SNR>=10
SNR<=15
?
SNR>15
SNR<=
27?
SNR>27
SNR<=
38?
SNR>38
?
Perform BPSK modulation
QPSK modulation
16 bit QAM modulation
64 bit QAM modulation
Calculate BER
Plot SNR vs BER
Assign throughput values based on modulation
Plot SNR vs THROUGHPUT

EXP NO:2 **DATE**:	**PERFORMANCE STUDIES OF ADAPTIVE MODULATION**

AIM:

To perform adaptive modulation and observe its characteristics using MATLAB.

SOFTWARE REQUIRED:

MATLAB R2022b

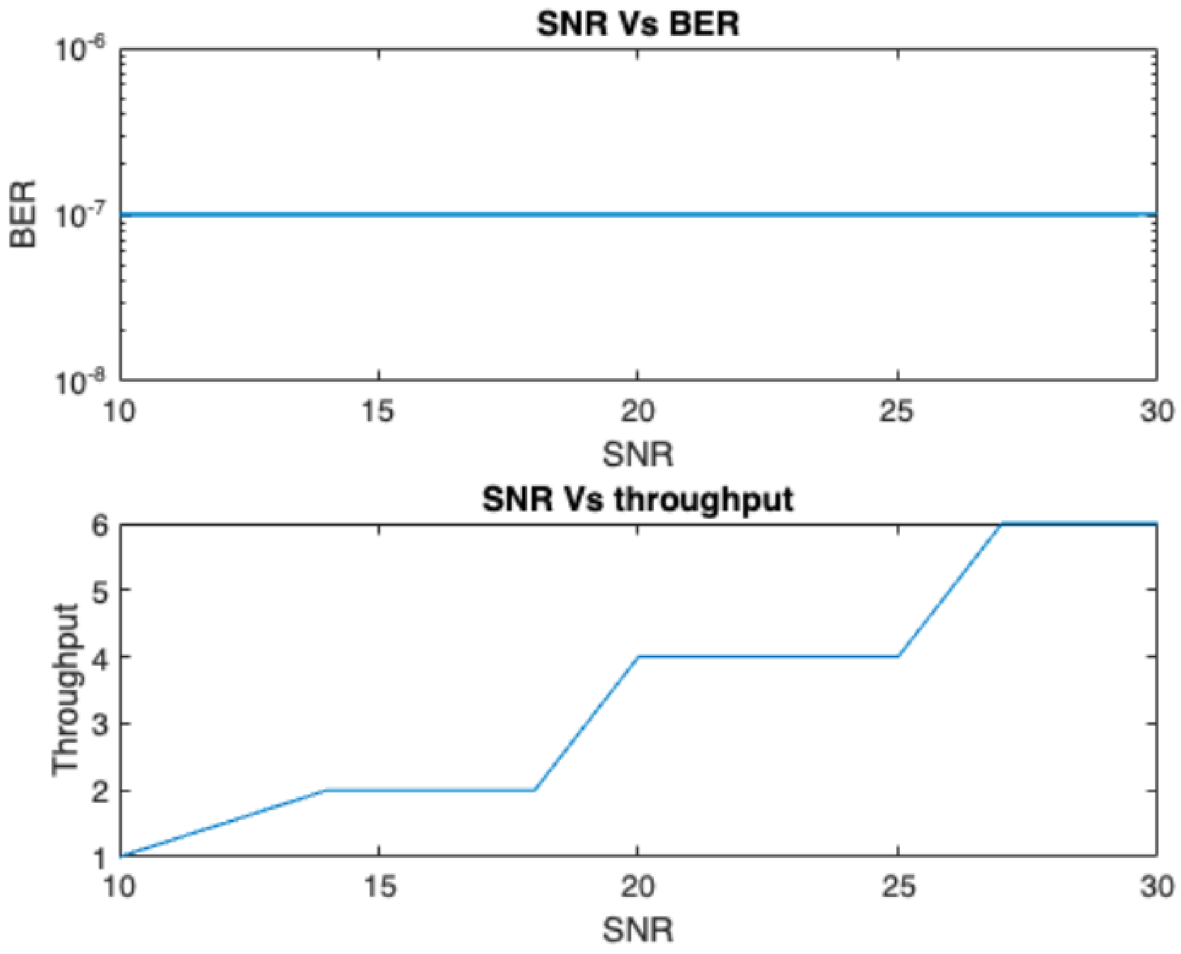
THEORY:

Adaptive Modulation is a technique which allows a radio to change its speed (Modulation rate) as conditions in the ratio network change. Interference from outside sources, such as changes in the environment (temperature free foliage, moving objects) all effect radio coverage. Adaptive modulation enables robust and spectrally efficient transmission over time varying channels. The basic premise is to estimate the channel at the receiver and feed this channel back to the transmitter, so that the transmission can be adapted relative to channel characteristics. Thus, the system can be designed efficiently for worst case channel characteristics.

ALGORITHM:

1. Define a random sequence of bits for Bpsk, Qpsk, 16 QAM,64 QAM
2. SNR range is also defined
3. If 10<=SNR<=15 use Bpsk Modulation
4. If 15<=SNR<=27 use Qpsk Modulation
5. If 27<=SNR<=38 use 16 QAM
6. Else use 64 QAM
7. If BER difference is zero, BER=10^-7
8. Plot the graph with x-axis :SNR and y-axis :BER

OUTPUT:


SNR Vs BER
BER
10^{-6}
10^{-7}
10^{-8}
10
15
20
25
30
SNR
SNR Vs throughput
Throughput
6
5
4
3
2
1
10
15
20
25
30
SNR

MATLAB CODE:

```
close all;
clc;
snr=randi([10,30],1,15);
snr=sort(snr);
err=[];
for i=1:length(snr)
if(snr(i)>=0 && snr(i)<13)
a=randi([0,1],1,1000); b=pskmod(a,2);
end
if(snr(i)>=13 && snr(i)<20)
a=randi([0,3],1,1000); b=pskmod(a,4);
end
if(snr(i)>=20 && snr(i)<26)
a=randi([0,15],1,1000); b=qammod(a,16);
end
if(snr(i)>=26 && snr(i)<30)
a=randi([0,63],1,1000); b=qammod(a,64);
end
f=awgn(b,snr(i),"measured");
if(snr(i)>=10 && snr(i)<13)
d=pskdemod(f,2);
end
if(snr(i)>=13 && snr(i)<20)
d=pskdemod(f,4);
end
if(snr(i)>=20 && snr(i)<26)
d=qamdemod(f,16);
end
if(snr(i)>=26 && snr(i)<30)
d=qamdemod(f,64);
end
end
[n,r]=biterr(a,d);
subplot(2,1,1);
semilogx(snr,err,"LineWidth",1);
xlabel("snr");
ylabel("ber");
title({"amc";"snr vs ber"});
thruput=[];
for (i=1:length(snr))
if(snr(i)>=10 && snr(i)<13)
thruput=[thruput 1];
end
if(snr(i)>=13 && snr(i)<20)
thruput=[thruput 2];
end
if(snr(i)>=20 && snr(i)<=25)
thruput=[thruput 4];
end
if(snr(i)>=26 && snr(i)<=30)
thruput=[thruput 6];
end
end
subplot(2,1,2);
plot(snr,thruput);
xlabel("snr");
ylabel("thruput");
title({"amc";"snr vs thruput"});
```

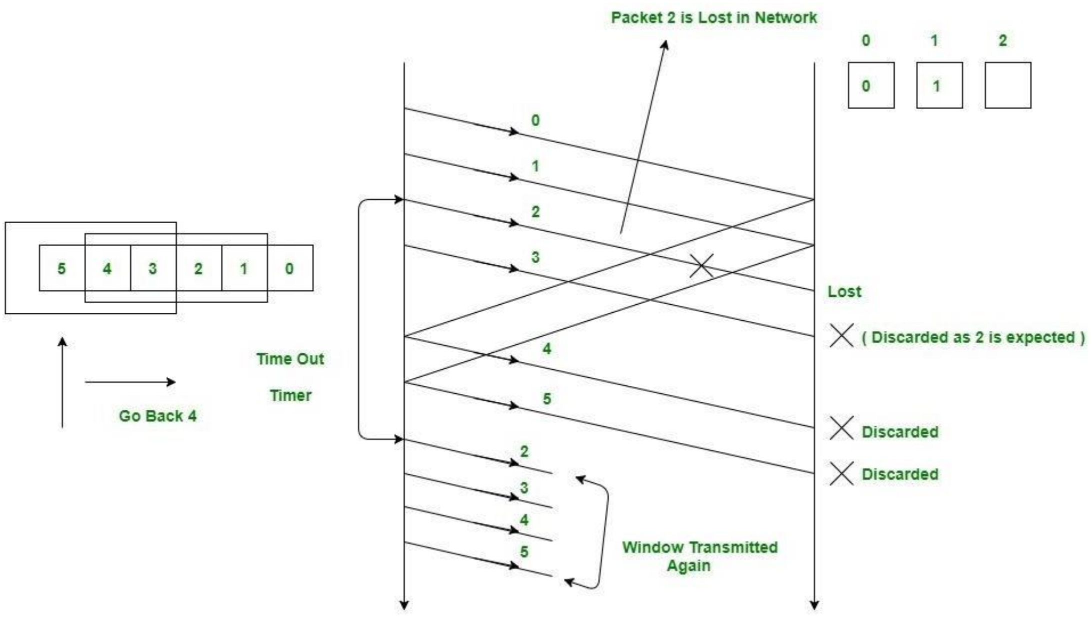
INFERENCE:

1. Thus, the different Modulation scheme is used for different SNR for lossless transmission of the signal.
2. When the SNR is low, BER is high and hence throughput is low.
3. When the SNR is high, BER is low and hence throughput is high.

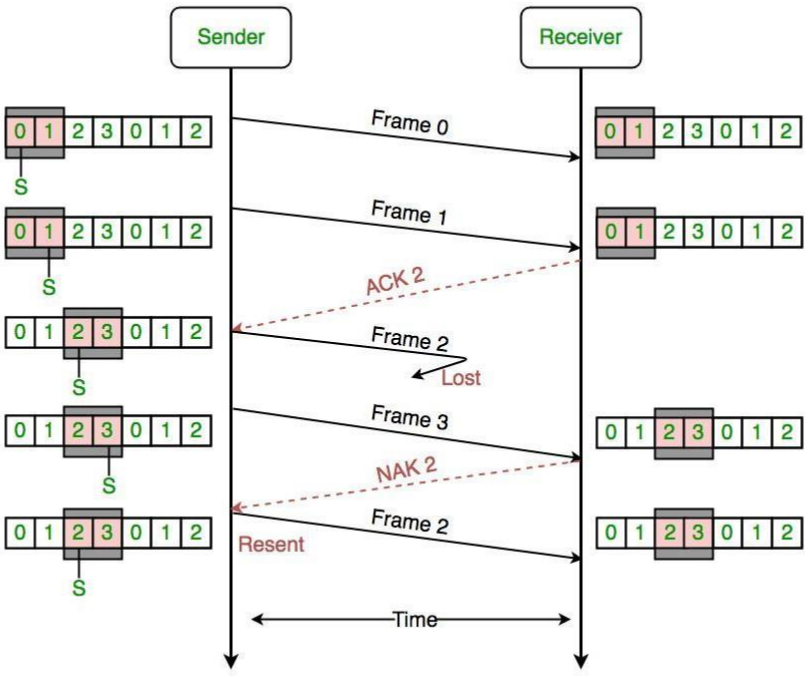
RESULT:

Thus, the Adaptive Modulation has been implemented and analysed using MATLAB.

GO BACK N PROTOCOL:



SELECTIVE REPEAT:



EXP.NO:3 **DATE:**	**PERFORMANCE STUDIES OF LLC PROTOCOLS**

AIM:

To implement:

1. Go Back ‘N’ Protocol
2. Selective Repeat Protocol

and to determine the throughput and delay for each.

SOFTWARE REQUIRED:

NETSIM SOFTWARE

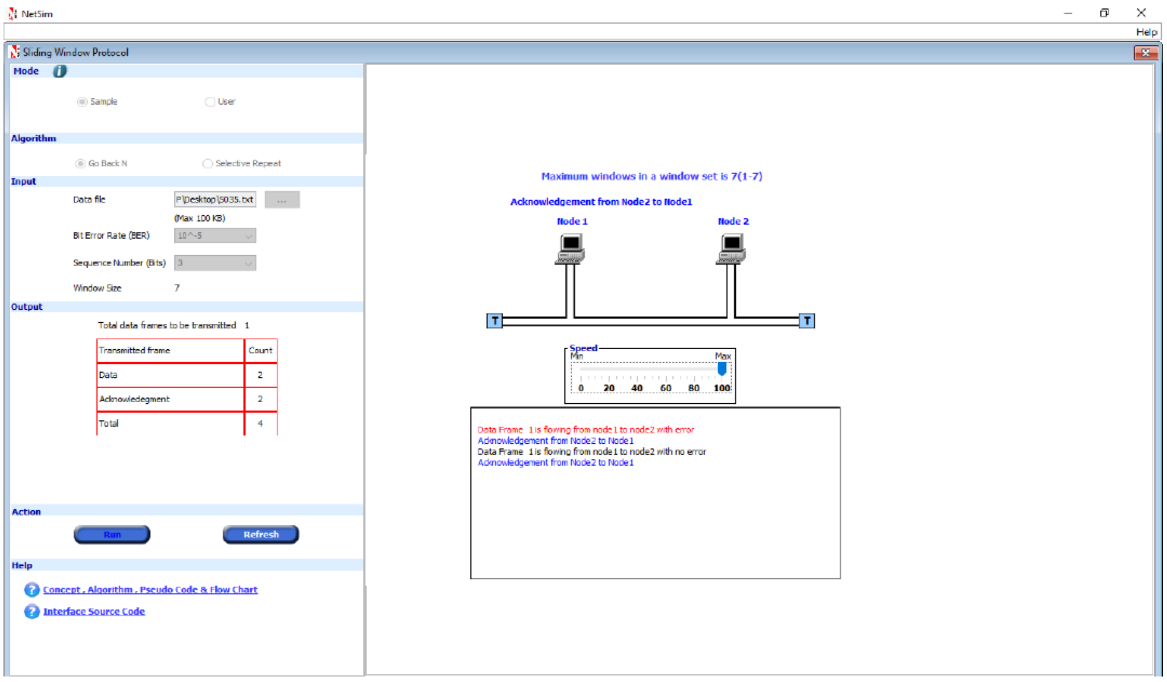
THEORY:

1) GO BACK ‘N’PROTOCOL:

Go Back ‘N’ is a connection-oriented transmission. The sender transmits the frames continuously. Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size. The sender has a window i.e., a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously. The size of the window depends on the protocol designer.

LAYOUT

GO BACK 'N' PROTOCOL:



Mode: [✓ Sample] User

Algorithm: [✓ Go Back N] Selective Repeat

Data file: P:\Desktop\5035.txt (Max 100 KB)

Bit Error Rate (BER): 10^-5

Sequence Number (Bits): 3

Window Size: 7

Total data frames to be transmitted 1

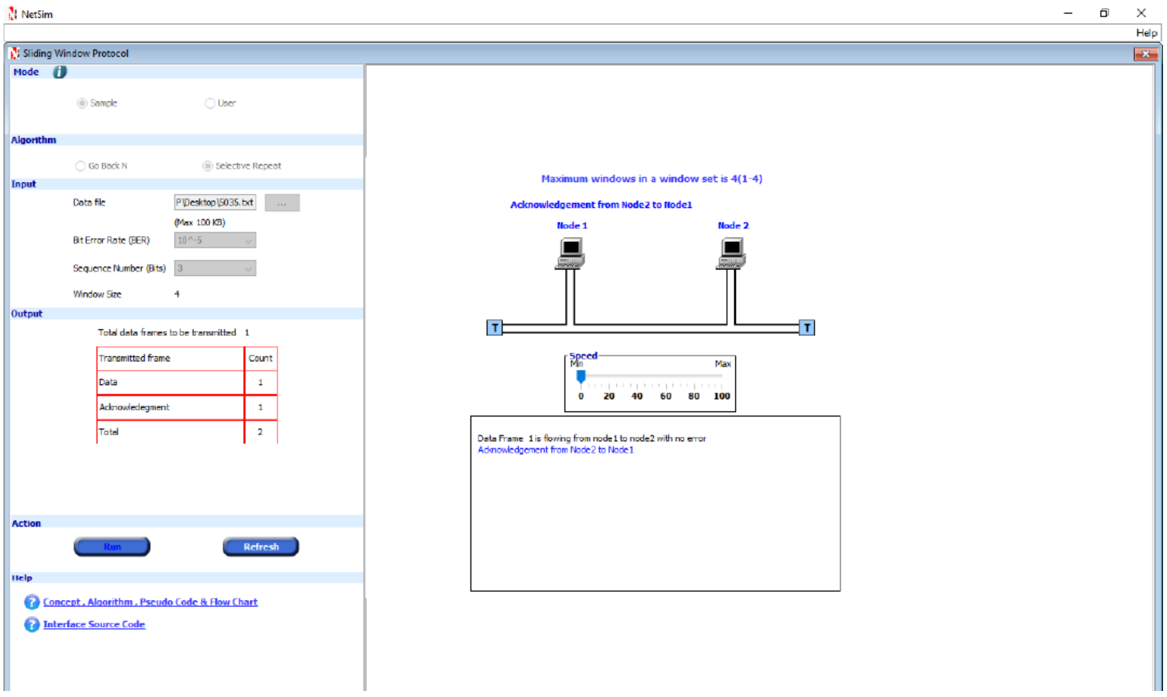
Transmitted frame	Count
Data	2
Acknowledgement	2
Total	4

Maximum windows in a window set is 7(1-7)

Acknowledgement from Node2 to Node1

```
Data Frame  1 is flowing from node1 to node2 with error
Acknowledgement from Node2 to Node1
Data Frame  1 is flowing from node1 to node2 with no error
Acknowledgement from Node2 to Node1
```

SELECTIVE REPEAT PROTOCOL:



Mode: [✓ Sample] User

Algorithm: Go Back N [✓ Selective Repeat]

Data file: P:\Desktop\5035.txt (Max 100 KB)

Bit Error Rate (BER): 10^-5

Sequence Number (Bits): 3

Window Size: 4

Total data frames to be transmitted 1

Transmitted frame	Count
Data	1
Acknowledgement	1
Total	2

Maximum windows in a window set is 4(1-4)

Acknowledgement from Node2 to Node1

```
Data Frame  1 is flowing from node1 to node2 with no error
Acknowledgement from Node2 to Node1
```

ALGORITHM:

1) The source code transmits the frames continuously.
2) Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size.
3) The source code has a window i.e., a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously.
4) The size of the window depends on the protocol designer.
5) For the first frame, the receiving node forms a positive acknowledgement if the frame is received without any error.
6) If subsequent frames are received without error (up to window size) cumulative positive acknowledgement is formed.
7) If the subsequent frame is received with error, the cumulative acknowledgement error-free frames are transmitted. If, In the same window two frames or more frames are received with error, the second and the subsequent error frames are neglected. Similarly, even the frames received without error after the receipt of a frame with error are neglected. The source code re-transmits all frames of window from the first error frame.
8) If the frames are errorless in the next transmission and if the acknowledgement is error-free, the window slides by the number of error-free frames being transmitted.
9) If the acknowledgement is transmitted with error, all the frames of window at source are retransmitted and window doesn’t slide.
10) This concept of replacing the transmission from the first error frame in the window is called as Go Back-N transmission flow control protocol.

TABULATION:

GO BACK "N" PROTOCOL:

GO BACK 'N' (Sequence number:3 bits) BER	**TOTAL DATA FRAMESTO BE TRANSMITTED**	**DATA**	**ACKNOWL EDGEMENT**	**TOTAL**
10^-5	3	2	2	4
10^-6	3	2	2	4
10^-7	3	1	1	2
10^-8	3	1	1	2
10^-9	3	1	1	2
NO ERROR	3	1	1	2

SELECTIVE REPEAT PROTOCOL:

SELECTIVE REPEAT (Sequence number:3 bits) BER	**TOTAL DATA FRAMESTO BE TRANSMITTED**	**DATA**	**ACKNOWLE DGEMENT**	**TOTAL**
10^-5	1	1	1	2
10^-6	1	1	1	2
10^-7	1	1	1	2
10^-8	1	1	1	2
10^-9	1	1	1	2
NO ERROR	1	1	1	2

2.SELECTIVE REPEAT PROTOCOL:

It is similar to Go Back ‘N’ Protocol, but the sender send frame only after the reception of ACK signal. It may be used as a protocol for delivery and ACK for message units for delivery of subdivided message. It is used as a protocol for delivery of message sender continuous to send frames specified by windows size even after becoming frameless. Once the sender has sent all the frame in its windows, it resends the frame number given by ACK and continues where it left off.

ALGORITHM:

1) The source node transmits the frames continuously.
2) Each frame in the buffer has a sequence number starting from 1and increasing up to the window size.
3) The source node has a window i.e., buffer to store the frames. This buffer size is the number of
    a. Frames to be transmitted continuously.
4) The receiver has a buffer to store the received frames. The size of the buffer depends upon the
    a. Window size defined by the protocol designer.
5) The size of the window depends according to the protocol designer.
6) The source node transmits frames continuously till the window size is exhausted. If any of the frames are received with error only those frames are requested for retransmission (with a negative acknowledgement).
7) If all the frames are received without error, a cumulative positive acknowledgement is sent.
8) If there is an error in frame 3, an acknowledgement for the frame 2 is sent and then only frame 3 is retransmitted. Now the window slides to get the next frames to the window.
9) If acknowledgement is transmitted with error, all the frames of window are retransmitted. Else ordinary window sliding takes place. (In implementation part, acknowledgement error is not considered).
10) If all the frames transmitted are errorless the next transmission is carried out for the new window.
11) This concept of repeating the transmission for the error frames only is called Selective Repeat transmission flow control protocol.

PROCEDURE:

1. Open NETSIM.
2. Click Programming
3. Transmission Flow Control.
4. Select Sample.
5. Select Go Back-N transmission.
6. Enter input data and BER.
7. Click Link to execute the program.
8. Repeat steps 1 to 3.
9. Select Selective Repeat protocol.
10. Enter input data and BER.
11. Click Link to execute the program.

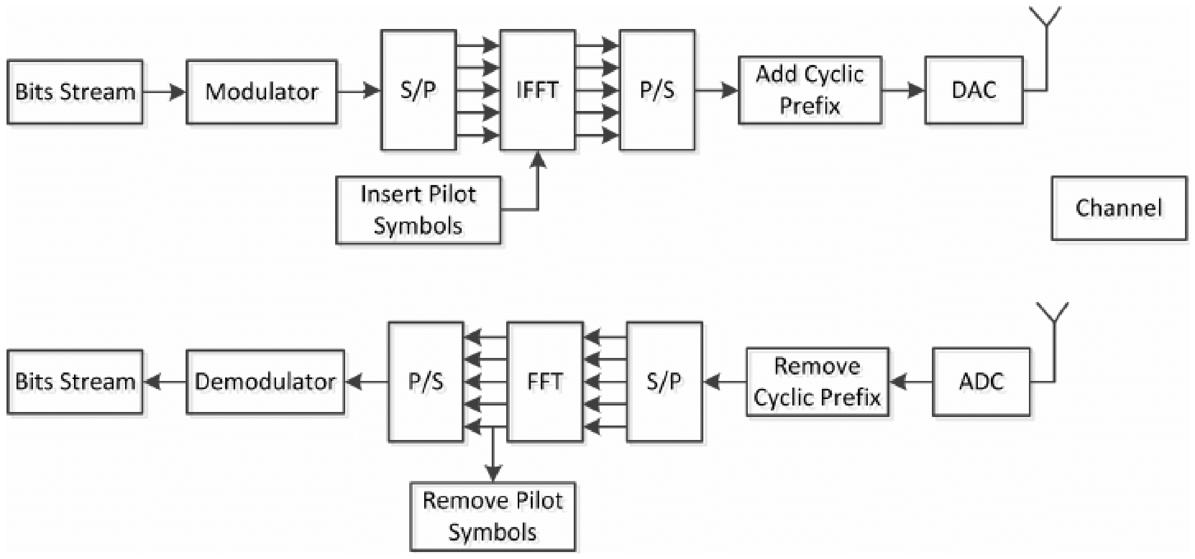
INFERENCE:

It is found that the Selective Repeat Protocol is the most optimum out of these two LLC protocols. But the complexity of the equipment is high. So, a trade-off exists between the total number of transmissions taken place and complexity.

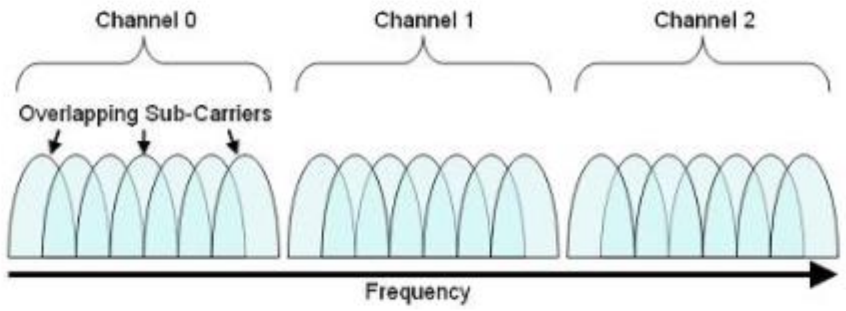
RESULT:

Thus, the LLC protocols such as “Go Back ‘N’” and “Selective Repeat” were studied using NETSIM and their performance were analyzed.

BLOCK DIAGRAM:



OFDM WAVEFORM:



EXP.NO:4 DATE:	**ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING**

AIM:

To implement multicarrier modulation technique (OFDM) using MATLAB.

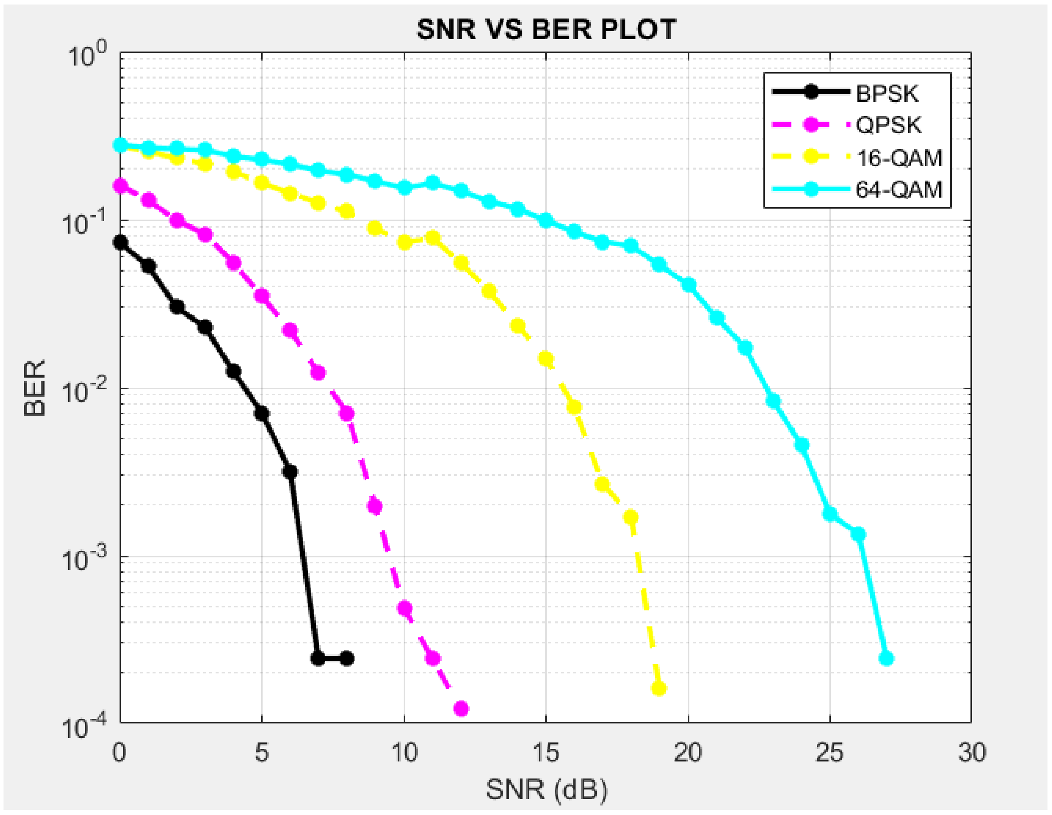
REQUIREMENTS:

- MATLAB 2023b

THEORY:

- Orthogonal Frequency Division Multiplexing (OFDM) is a digital multi-carrier modulation scheme that extends the concept of single subcarrier modulation by using multiple subcarriers within the same single channel. Rather than transmit a high-rate stream of data with a single subcarrier, OFDM makes use of a large number of closely spaced orthogonal subcarriers that are transmitted in parallel.
- Each subcarrier is modulated with a conventional digital modulation scheme (such as QPSK, 16QAM, etc.) at low symbol rate. However, the combination of many subcarriers enables data rates similar to conventional single-carrier modulation schemes within equivalent bandwidths.
- The OFDM signal can be described as a set of closely spaced FDM subcarriers. In the frequency domain, each transmitted subcarrier results in a sinc function spectrum with side lobes that produce overlapping spectra between subcarriers.
- At orthogonal frequencies, the individual peaks of subcarriers all line up with the nulls of the other subcarriers. This overlap of spectral energy does not interfere with the system’s ability to recover the original signal. The receiver multiplies (i.e., correlates) the incoming signal by the known set of sinusoids to recover the original set of bits sent.
- The use of orthogonal subcarriers allows more subcarriers per bandwidth resulting in an increase in spectral efficiency. In a perfect OFDM signal, Orthogonality prevents interference between overlapping carriers

MATLAB OUTPUT:



ALGORITHM:

- Generate the data points.
- Modulate the data.
- Convert columns to row(serial to parallel)
- Take IFFT for data in the matrix.
- Add cyclic prefix to create actual OFDM block.
- Connect rows to column (parallel to serial).
- Transmit the OFDM Signal.
- Received signal is converted to parallel.
- FFT is performed after removing cyclic prefix.
- Convert the data to serial stream.
- Demodulate the data.
- Repeat it for different SNRs
- Plot the BER vs SNR graph

MATLAB CODE:

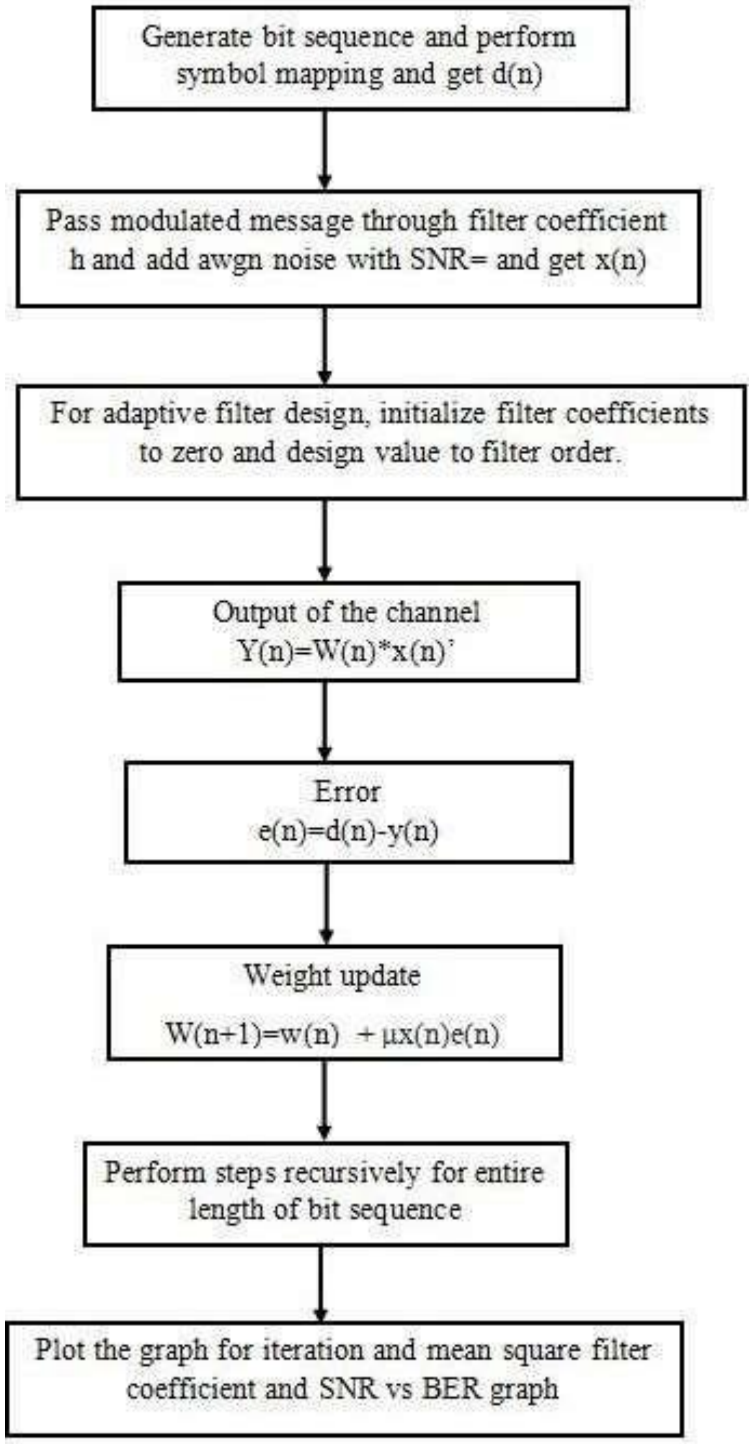
```
clc;
close all;
x = randi([0 1], 1, 4096);
snr = 0:30;
biterror = zeros (4, length (snr));

for i = 1:4
    y = pskmod(x, 2^i);
    if i == 3
        y = qammod(x, 2^ (i+1));
    end
    if i == 4
        y = qammod(x, 2^ (i+2));
    end
    p = reshape(y, 64, 64);
    q = ifft (p, 64);
    s = reshape (q, 1, 4096);
    be = zeros (1, length(snr));
```

```
for j = 1:length(snr)
        % Channel model
        h = 2+3i;
        r = h * s;
        n = awgn(r, snr(j), 'measured');
        m = inv(h) * n;
        p11 = reshape(m, 64, 64);
        q11 = fft(p11, 64);
        s11 = reshape(q11, 1, 4096);
        y11 = pskdemod(s11, 2^i);
        if i == 3
            y11 = qamdemod(s11, 2^(i+1));
        end
        if i == 4
            y11 = qamdemod(s11, 2^(i+2));
        end
[num1, e1] = biterr(y11, x)

 be(j) = e1;
    end
    biterror(i, :) = be;
end

semilogy(snr, biterror(1,:), '*-k', 'linewidth', 2); hold on;
semilogy(snr, biterror(2,:), '*--m', 'linewidth', 2); hold on;
semilogy(snr, biterror(3,:), '*--y', 'linewidth', 2); hold on;
semilogy(snr, biterror(4,:), '*-c', 'linewidth', 2); hold on;

xlabel('SNR (dB)');
ylabel('BER');
title('SNR VS BER PLOT');
legend('BPSK', 'QPSK', '16-QAM', '64-QAM');
grid on;
```

RESULT:

Thus, implementation of multicarrier modulation (Orthogonal Frequency Division Multiplexing) is carried out using MATLAB.

FLOWCHART: (LMS)


Generate bit sequence and perform symbol mapping and get d(n)
Pass modulated message through filter coefficient h and add awgn noise with SNR= and get x(n)
For adaptive filter design, initialize filter coefficients to zero and design value to filter order.
Output of the channel Y(n)=W(n)*x(n)'
Error e(n)=d(n)-y(n)
Weight update W(n+1)=w(n) + µx(n)e(n)
Perform steps recursively for entire length of bit sequence
Plot the graph for iteration and mean square filter coefficient and SNR vs BER graph

EXP NO: 5 **DATE:**	**PERFORMANCE OF EQUALIZERS**

AIM:
To implement equalization techniques for wireless channels using Least Mean Squares (LMS) and Zero Forcing algorithms.

SOFTWARE REQUIRED:

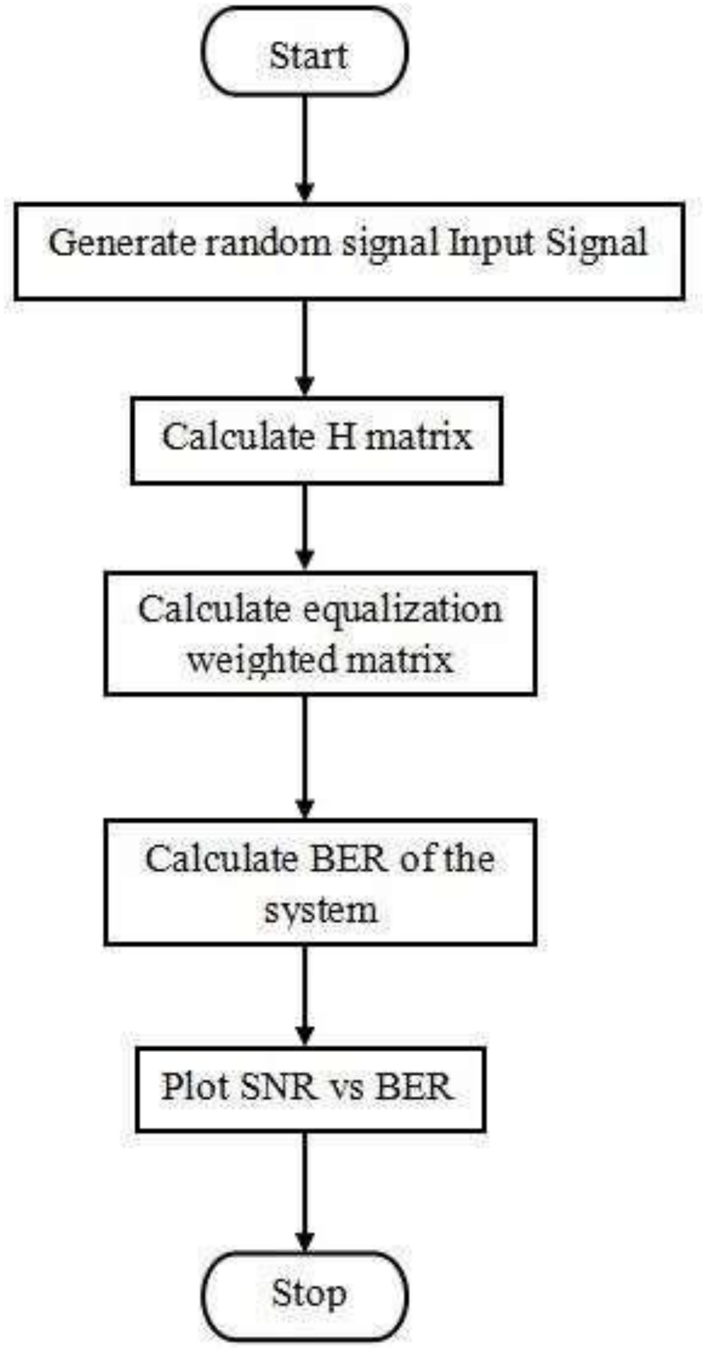
MATLAB 2022b

THEORY:
A channel equalizer is an important component of a communication system and is used to mitigate the ISI (inter symbol interference) introduced by the channel. The equalizer depends upon the channel characteristics. These are usually employed to reduce the depth and duration of the fades experienced by a receiver in a local area which are due to motion. An equalizer within a receiver compensates for the average range of expected channel amplitude and delay characteristics.

Least Mean Squares (**LMS**) algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean square of the error signal (difference between the desired and the actual signal). It is a stochastic gradient descent method in that the filter is only adapted based on the error at the current time.

Zero Forcing Equalizers is a linear equalizer which uses unit impulse response to compensate for channel effect. An overall impulse response is equal to one for the defected symbol and zero for all other received symbols. The noise may be increased in this process.

FLOWCHART: (Zero forcing Equalizers)


Start
Generate random signal Input Signal
Calculate H matrix
Calculate equalization weighted matrix
Calculate BER of the system
Plot SNR vs BER
Stop

LEAST MEAN SQUARES (LMS) ALGORITHM

In the Least Mean Squares (LMS) algorithm:

Initialization:
- Start with the filter coefficients set to some initial values. Typically, they are set to zero or small random values.

Iterative Update:
- At each iteration:
  - Calculate the error between the desired signal and the filter output.
  - Update each filter coefficient using the error, the input signal, and a learning rate.

Filter Output:
- The filter output is computed by convolving the filter coefficients with the input signal.

Convergence:
- The algorithm iterates until convergence criteria are met, such as reaching a specified number of iterations or when the change in the filter coefficients falls below a predefined threshold.

Adaptation Speed:
- The learning rate controls how quickly the algorithm adapts to changes in the input signal. A higher learning rate leads to faster convergence but may cause instability, while a lower learning rate provides stability but may converge slower.

Applications:
- LMS finds applications in various areas such as adaptive equalization in communication systems, echo cancellation in telecommunication, noise cancellation in audio processing, and system identification in control systems.

The simplicity and effectiveness of the LMS algorithm make it widely used in adaptive signal processing tasks.

$$y_k = h_0 x_k + h_1 x_{k-1} + n_0$$

$$y_{k+2} = h_0 x_{k+2} + h_1 x_{k+1} + n_2$$

$$y_{k+1} = h_0 x_{k+1} + h_1 x_k + n_1$$

$$\begin{bmatrix} y_{k+2} \\ y_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & 0 & 0 \\ 0 & h_0 & h_1 & 0 \\ 0 & 0 & h_0 & h_1 \end{bmatrix} . \begin{bmatrix} x_{k+2} \\ x_{k+1} \\ x_k \\ x_{k-1} \end{bmatrix} + \begin{bmatrix} n_2 \\ n_1 \\ n_0 \end{bmatrix}$$

$$\overline{1_2} = C^T H = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

$$C = (HH^T)^{-1} . \overline{1_2}$$

Where;

- y→ Output sequence at instance *k*
- x→ Input sequence at instant *k*
- n→ Noise sequence at instant *k*
- $h0$ $h1$ → Equalizing filter coefficients
- C→ Zero Forcing Constant

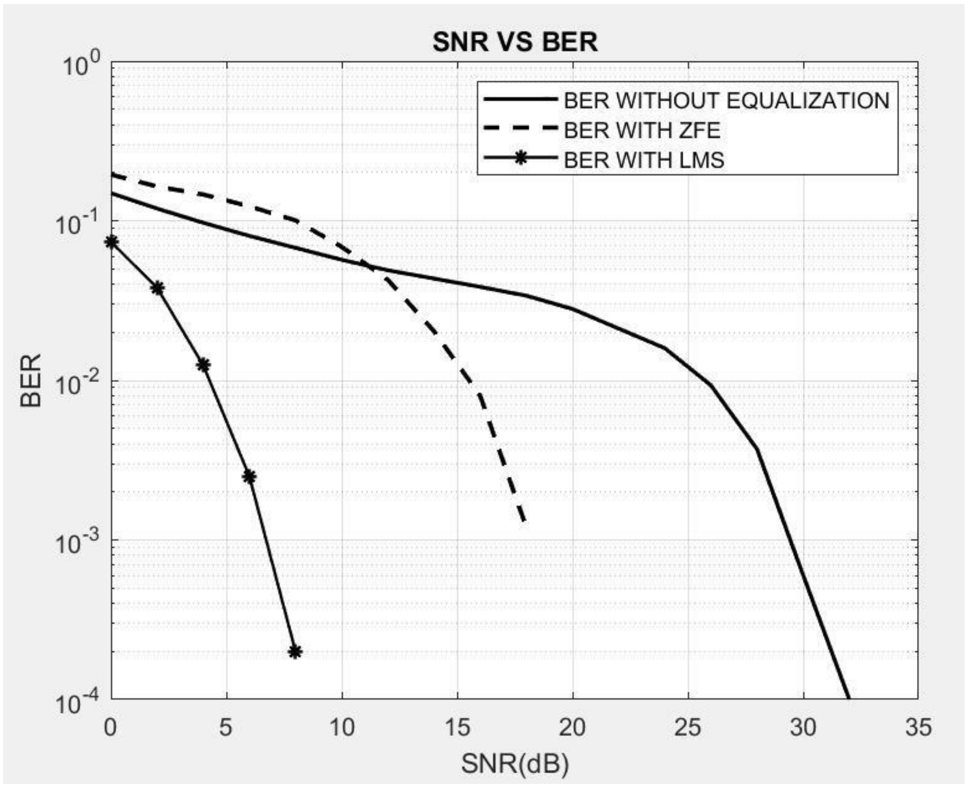
ZERO FORCING EQUALIZERS

Zero Forcing Equalization aims to design a filter that eliminates intersymbol interference, ensuring the received signal perfectly reconstructs the transmitted symbols without interference from adjacent symbols.

MATLAB CODE:

```
clc; clear all; close all;
message =
randi([0,1],1,10000);snr =
0:2:40; mod = 2;
L = 2; r = 3; % 3 TAP EQUALISER
modulated_bpsk_msg =
pskmod(message,mod);h0 = 1; h1
= 0.7;
H = zeros(r,r+L-1); % ORDER OF H MATRIX
% FORMING H MATRIX FOR 3 TAP EQUALISER
for p = 1:r
H(p,p:p+L-1) = [h0 h1];
end
x = []; % FORMING INPUT MATRIX X
X = modulated_bpsk_msg; X_1 =
circshift(X,1);X_1(1) = 0; X1 =
circshift(X,-1);
X1(end) = 0; X2 =
circshift(X,-2); X2(end-
1:end)= 0; x =
[X2;X1;X;X_1];
C = ((H*H')\H)*[0;0;1;0];
ber_without_ZFE = [];
ber_with_ZFE = [];for p =
1:length(snr)
y = awgn([h0
h1]*[X;X_1],snr(p),'measured');Noise
= y - ([h0 h1]*[X;X_1]);
Noise_1 = circshift(Noise,-1);
Noise_1(end) = 0; Noise_2 =
circshift(Noise,-2);Noise_1(end-1:end) =
0;
Y = (H*x)+ [Noise_2;Noise_1;Noise]; X_PRIME = C.'*Y;
```

MATLAB OUTPUT:



```
Demodulated_BPSK_msg_with_ZFE
=pskdemod(X_PRIME',mod);
Demodulated_BPSK_msg_without_ZFE =
pskdemod(y',mod); [number1,ratio1] =
biterr(message,Demodulated_BPSK_msg_with_ZFE');
[number2,ratio2] =
biterr(message,Demodulated_BPSK_msg_without_ZFE');
ber_with_ZFE = [ber_with_ZFE,ratio1];
ber_without_ZFE =
[ber_without_ZFE,ratio2];end
semilogy(snr,ber_with_ZFE,'-
k','LineWidth',1.6) hold on
semilogy(snr,ber_without_ZFE,'—
k','LineWidth',1.8) hold on
n = length(message); M = 25; w = zeros(1,M);
wi = zeros(1,M); E = []; mu = 0.0005;
msg_bpsk = pskmod(message,mod);
ber_without_LMS = []; ber_with_LMS = [];
for k = 1:length(snr)
msg_rx = awgn(msg_bpsk, snr(k), 'measured');
for i = M:n
E(i) = msg_bpsk(i) - wi*msg_rx(i:-1:i-M+1)';
wi = wi + 2*mu*E(i)*msg_rx(i:-1:i-M+1);
end
msg_eq = zeros(n,1); for i = M:n
j = msg_rx(i:-1:i-M+1); msg_eq(i) = ((wi)*(j)');
end
Demod_with_LMS = pskdemod(msg_eq,mod)';
Demod_without_LMS =
pskdemod(msg_rx,mod); [n1,r1] =
biterr(message,Demod_with_LMS); [n2,r2] =
biterr(message,Demod_without_LMS);
ber_without_LMS = [ber_without_LMS,r1];
ber_with_LMS = [ber_with_LMS,r2];
end
grid on; semilogy(snr,ber_with_LMS,'-
*k','LineWidth',1.2) title('SNR VS BER');
xlabel('SNR(dB)');ylabel('BER'); legend('BER
WITHOUT EQUALIZATION','BER WITH
ZFE','BER WITH LMS');
```

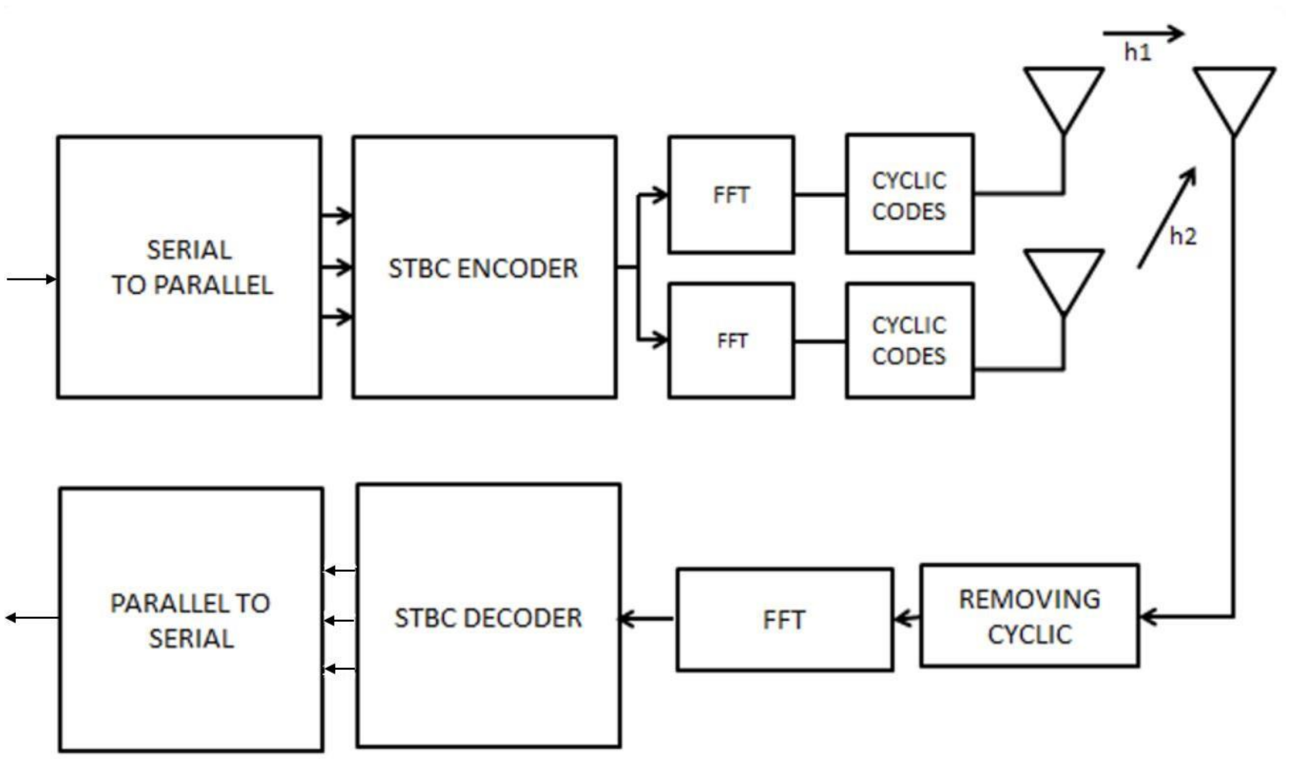
INFERENCE:

1. In the Least Mean Squares (LMS) algorithm of equalization techniques of wireless channels, when the epoch iterations increase, error decreases thereby decreasing the Mean Squared Error (MSE).
2. In the case of Zero Forcing Equalizer, it can be observed that it nullifies inter-symbol interference (ISI) in the modulated signals. But noise obtained in the process increases accordingly.
3. Hence, comparing both the algorithms, we can observe that Least Mean Squares (LMS) algorithm is efficient than that of Zero Forcing Equalization technique in terms of noise performance.

RESULT:

Hence, channel equalization using the Least Mean Squares (LMS) algorithm andZero Forcing Equalizer are successfully performed using MATLAB and the cost function/BER-SNR graphs for the same are plotted.

BLOCK DIAGRAM:



EXP NO: 6 **DATE:**	**SPACE TIME BLOCK CODES**

AIM:

To perform space time block codes (Alamouti coding) for a system that has two transmitter and one receiver.

SOFTWARE REQUIRED:

MATLAB 2022b

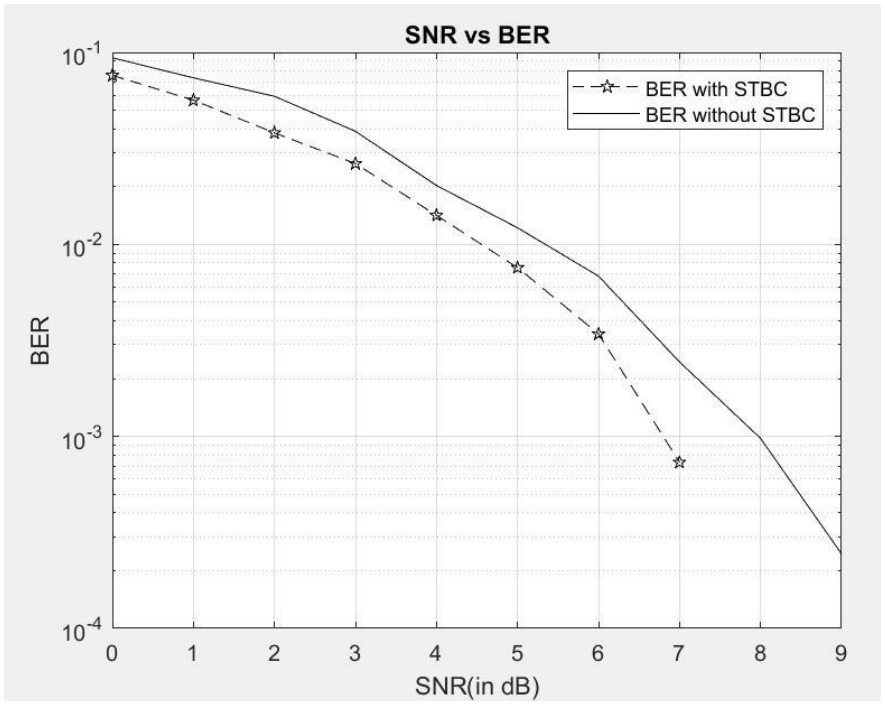
THEORY:

Space-time block codes are used for MIMO systems to enable the transmission of multiple copies of a data stream across a number of antennas and to exploit the various received versions of the data to improve the reliability of data-transfer. Space-time coding combines all the copies of the received signal in an optimal way to extract as much information from each of them as possible.

A space time block code is usually represented by a matrix. Each row represents a time slot and each column represents one antenna's transmissions over time. Space-time block codes (STBC) are a generalized version of Alamouti scheme. These schemes have the same key features. Therefore, these codes are orthogonal and can achieve full transmit diversity specified by the number of transmit antennas. In another word, spacetime block codes are a complex version of Alamouti's space- time code, where the encoding and decoding schemes are the same as there in the Alamouti space-time code in both the transmitter and receiver sides.

At the receiver side, when signals are received, they are first combined and then sent to the maximum likelihood detector where the decision rules are applied.

MATLAB OUTPUT:



ENCODING:

It was designed for a two-transmit antenna system and has the coding matrix:

$$C_2 = \begin{bmatrix} c_1 & c_2 \\ -c_2^* & c_1^* \end{bmatrix}$$

Where * denotes complex conjugate.

It is readily apparent that this is a rate-1 code. It takes two time slots to transmit two symbols. Using the optimal decoding scheme discussed below, the Bit Error Rate (BER) of this STBC is equivalent to 2 n R – branch maximal ratio combining (MRC). This is a result of the perfect orthogonality between the symbols after receive processing – there are two copies of each symbol transmitted and n R copies received.

DECODING:

One particularly attractive feature of orthogonal STBCs is that maximum likelihood decoding can be achieved at the receiver with only linear processing. To consider a decoding method, a model of the wireless communications system is needed.

ALGORITHM:

1. Generate random binary sequence of +1s and -1s.
2. Group them into pairs of two symbols.
3. Code it as per the Alamouti Space Time Code, multiply the symbols with the channels and then add AWGN.
4. Equalize the received symbols.
5. Perform hard decision decoding and count the bit errors.
6. Repeat for multiple values and plot the simulation.

MATLAB CODE:

```
clc; clear all; close all;

n = randi([0,1],1,4096);

a = reshape(n,length(n),1); bpskmod = pskmod(a,2);

snr = 0:1:30;

h1 = 2+1j; h2 = 1-2j;

y = []; M= []; Op = [];

OpwoutSTBC = [];

q = 1;

for l = 1:length(snr) for p = 1:2:length((n))-1

c1 = (h1*bpskmod(p,1))+(h2*bpskmod(p+1,1)); M(p,q) = awgn(c1,snr(l),'measured');

c2 = (-h1*conj(bpskmod(p+1,1)))+(h2*conj(bpskmod(p,1))); M(p+1,q) =
awgn(c2,snr(l),'measured');

end

for r = 1:2:(length(n) -1 )

y(r,q) = (conj(h1)*M(r,1)) + h2*conj(M(r+1,1));

y(r+1,q) = (conj(h2)*M(r,1)) - h1*conj(M(r+1,1)); end

t1 = pskdemod(y,2);

Op(l,:) = reshape (t1,1,length(n)); [number,ratio] = biterr(Op,n);

end

semilogy(snr,ratio,'pentagram--C','Color','black');

xlabel("SNR(in dB");

ylabel("BER"); hold on

%without STBC

for l = 1:length(snr)

rec = awgn((h1*bpskmod),snr(l),"measured"); demod = pskdemod(rec,2);

OpwoutSTBC(l,:) = reshape(demod,1,length(n));
```

```
[number1,ratio1] = biterr(OpwoutSTBC,n);
end
semilogy(snr,ratio1,'-k');
 grid on
legend('BER with STBC','BER without STBC');
 xlabel("SNR(in dB");
ylabel("BER");
title("SNR vs BER");
```

INFERENCE:

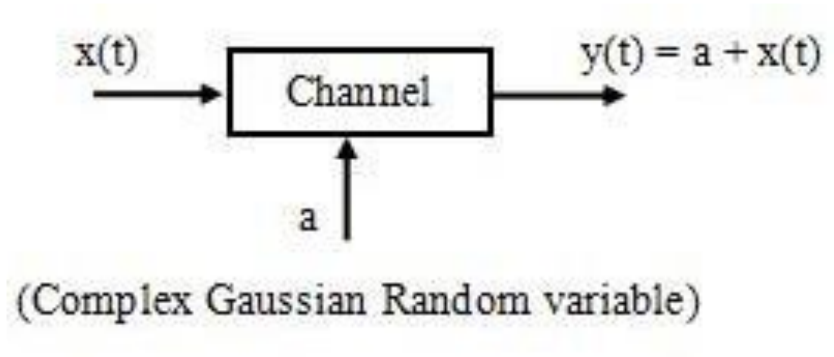
- No need of Channel information since the transmitter antenna is independent of Channel State
- Since it requires two Transmit antennas, this scheme consumes more power.

RESULT:

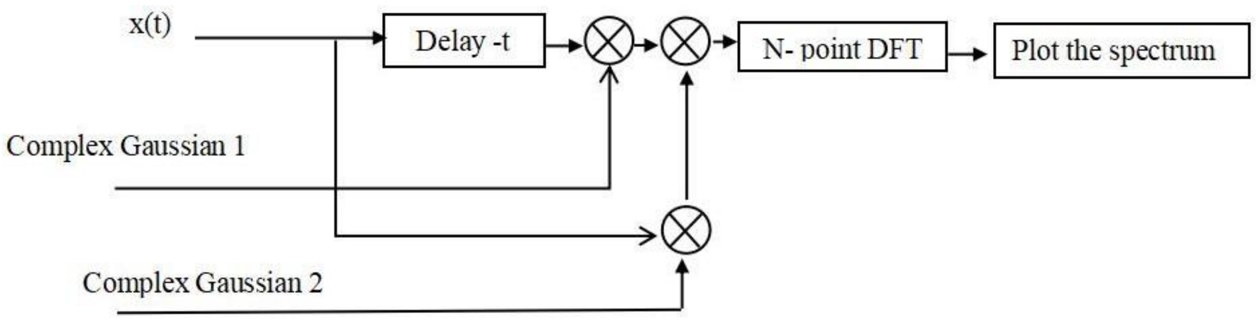
Thus, the Space Time Block Code using Alamouti coding scheme is studied, and the Bit Error Rate is analyzed.

BLOCK DIAGRAM:

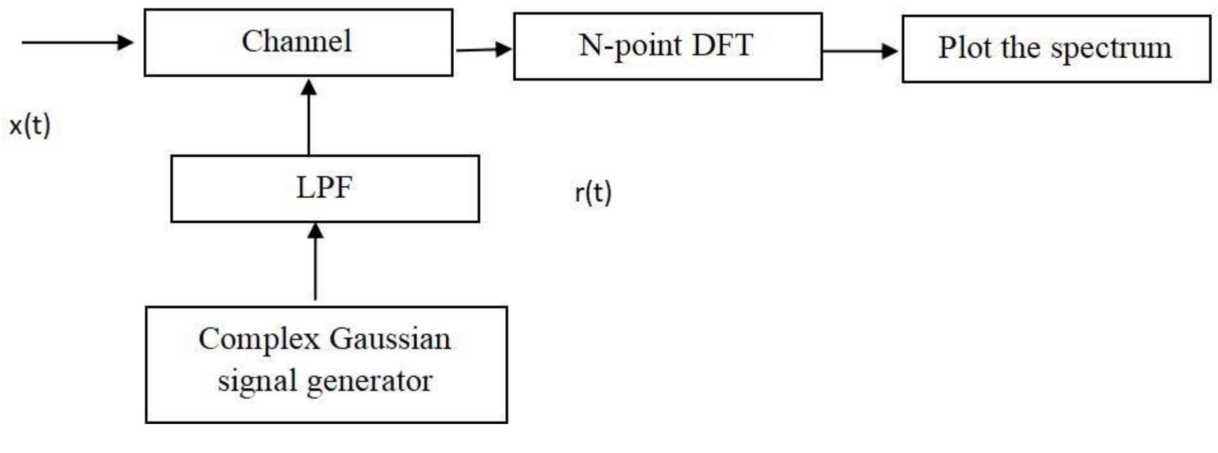
- **Slow and Flat Fading**



- **Slow and Frequency Selective Fading**



- **Fast and Flat Fading**



EXP NO: 7 **DATE:**	**Characterization of Wireless Fading Channels**

AIM
To plot the spectra of two-tone signal due to slow flat fading, slow frequency selective fading,fast flat fading, fast frequency selective fading, also to analyse the BER performance of Rayleighand Rician channel.

SOFTWARE REQUIRED

MATLAB R2022b

THEORY
Based on multipath delay spread there are two types of small scale fading viz. flat fading and frequency selective fading. These multipath fading types depend on propagation environment.

FLAT FADING
The wireless channel is said to be flat fading if it has constant gain and linear phase response over a bandwidth which is greater than the bandwidth of the transmitted signal. In thistype of fading all the frequency components of the received signal fluctuate in same proportionssimultaneously. It is also known as non-selective fading

- Signal BW > Channel BW
- Symbol Period > Delay Spread

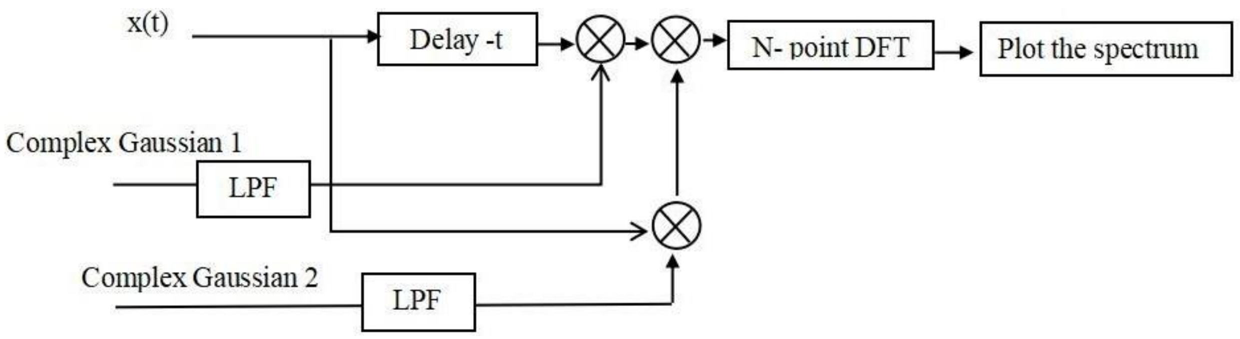
The effect of flat fading is seen as decrease in SNR. These flat fading channels are known as amplitude varying channels or narrow band channels.

FREQUENCY SELECTIVE FADING

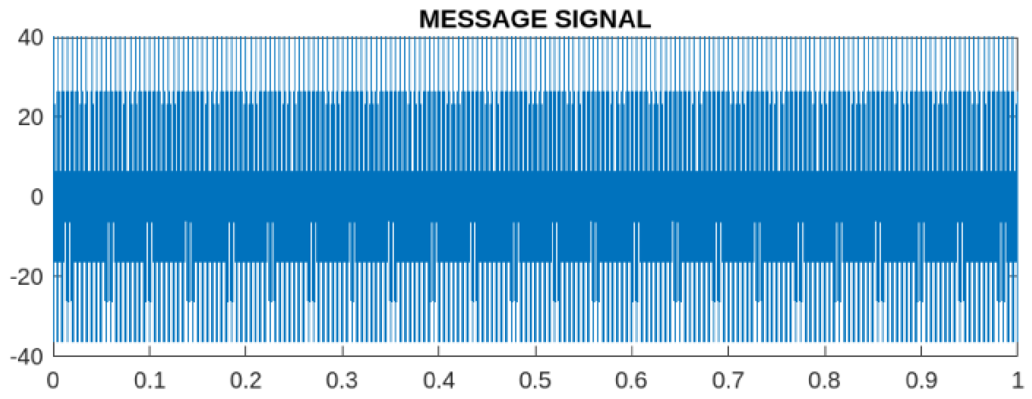
It affects different spectral components of a radio signal with different amplitudes. Hence, the name selective fading

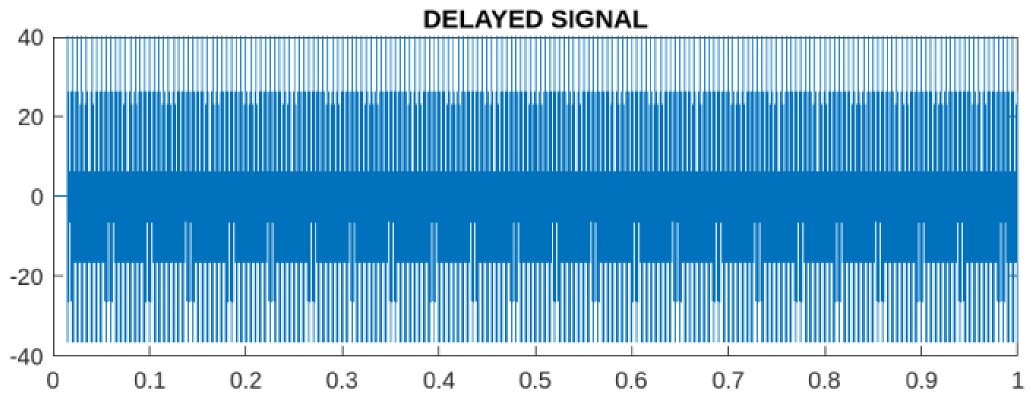
- Signal BW > Channel BW
- Symbol period < Delay Spread

Fast and Frequency Selective Fading



MATLAB OUTPUT:





Based on doppler spread there are two types of fading viz. fast fading and slow fading. Thesedoppler spread fading types depend on mobile speed i.e. speed of receiver with respect to transmitter.

FAST FADING

The phenomenon of fast fading is represented by rapid fluctuations of signal over small areas (i.e. bandwidth). When the signals arrive from all the directions in the plane, fast fading willbe observed for all direction so motion. Fast fading occurs when channel impulse response changesvery rapidly within the symbol duration.

High doppler spread

- Symbol period > Coherence time
- Signal Variation < Channel variation

These parameters result into frequency dispersion or time selective fading due to doppler spreading. Fast fading is result of reflections of local objects and motion of objects relative to thoseobjects.

SLOW FADING

Slow fading is result of shadowing by buildings, hills, mountains and other objects over the path.

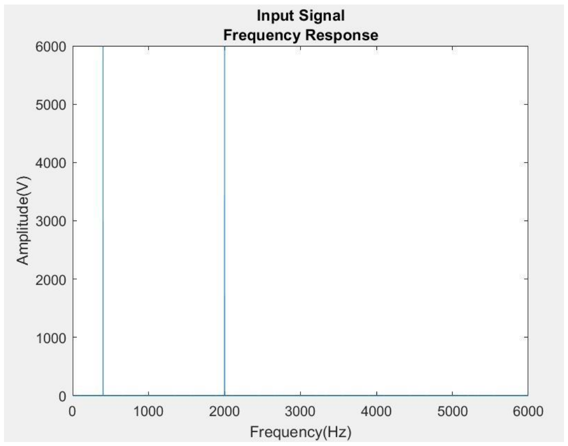
- Low Doppler Spread
- Symbol period << Coherence time
- Signal Variation >Channel Variation

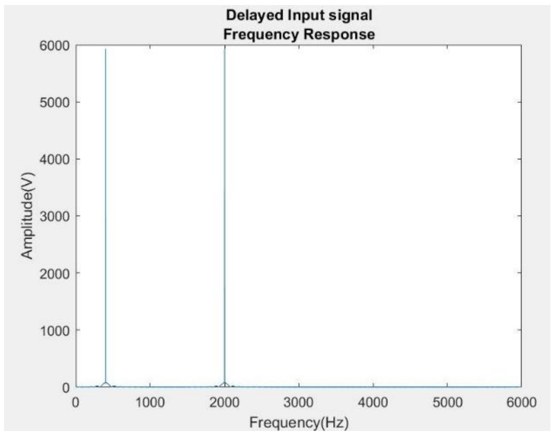
Slow fading results in a loss of SNR. Error correction coding and receiver diversity techniquesare used to overcome effects of slow fading.

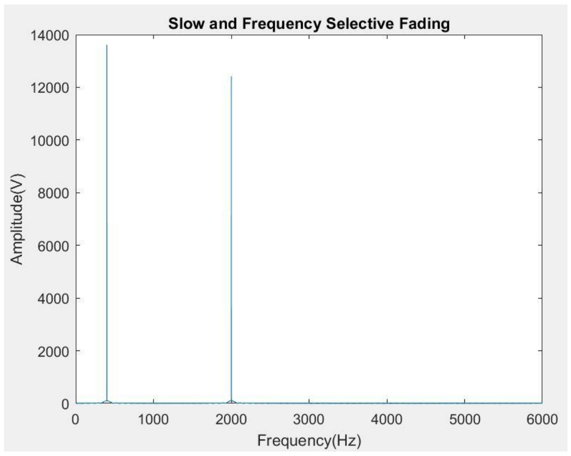
ALGORITHM:

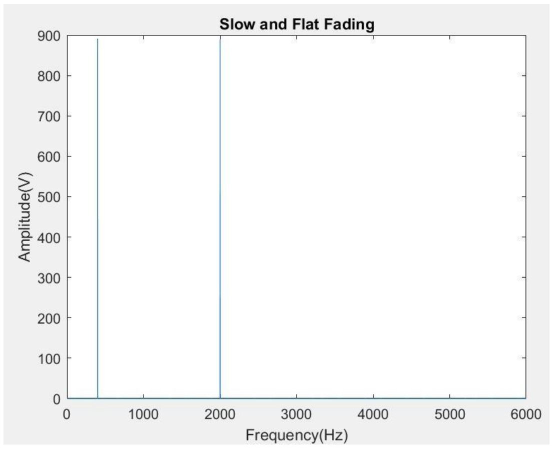
1. Generate a two-tone message signal.
2. Fix sampling frequency.
3. Generate a delayed signal.
4. Find the FFT of message signal.
5. For slow & flat fading, generate random complex gaussian variable and multiply with message signal.
6. Multiply one of the variables with message and the other variable with delayed signal.
7. For fast& flat fading, generate a random complex Gaussian variable and pass itthrough low pass filter and multiply it with message signal.
8. For fast & frequency selective fading, generate two random complex Gaussian variables. Multiply one of the variables with message and the other variable with delayed signal. Pass it through lowpass filter and multiply it with message signal.Find the FFT of signal and plot it.

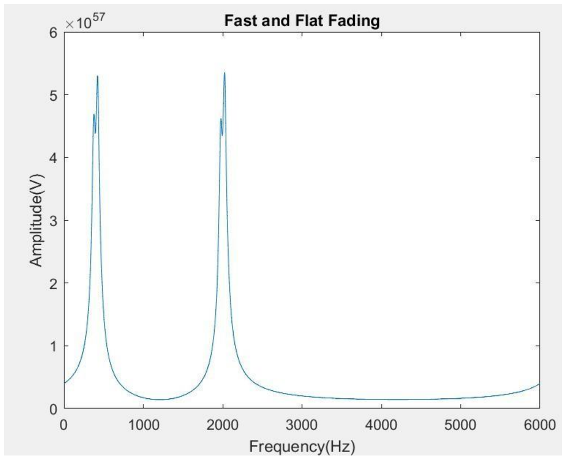
MATLAB OUTPUT:

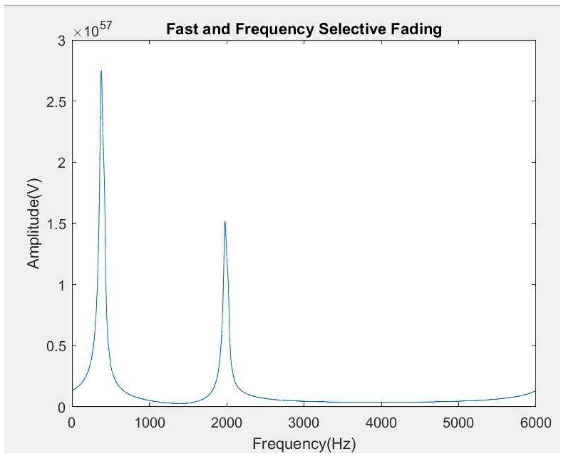










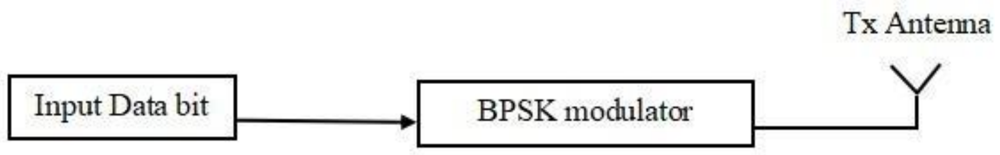


MATLAB CODE

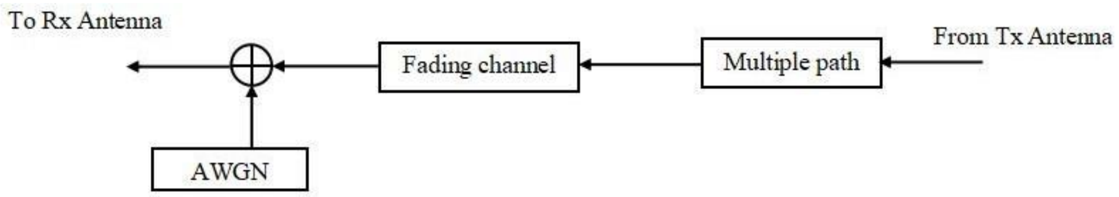
```
clc, clear all; close all;
fs = 6000; f1 = 2000; f2 = 400;
ts = 0:(1/fs):1-(1/fs);
signal1 = exp(complex(0,2*pi*f1*ts)); signal2 = exp(complex(0,2*pi*f2*ts));
message = signal1+signal2;
delayed = [zeros(1,77) message(1:length(message)-77)];
%FREQUENCY RESPONSE
freqres = fft(message); figure; magnitudeplot = abs(freqres); plot(1:fs,magnitudeplot);
title({'Input Signal';'Frequency Response'}); xlabel('Frequency(Hz)');
ylabel('Amplitude(V)'); figure; freqres2=fft(delayed); plot(1:fs,abs(freqres2));
title({'Delayed Input signal';'Frequency Response'}); xlabel('Frequency(Hz)');
ylabel('Amplitude(V)');
%Slow and Flat Fading
h = randn + (1i*randn); y1 = message.*h; freqres = fft(y1);
magnitudeplot = abs(fft(y1)); figure; plot(1:fs,magnitudeplot(1:fs)); title('Slow and
Flat Fading');
xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');
%Slow and Frequency Selective Fading h1 = randn + (1i*randn);
h2 = randn + (1i*randn);
trans2 = (h1.*message) + (h2.*delayed); magnitudeplot = abs(fft(trans2)); figure;
plot(1:fs,magnitudeplot);
title('Slow and Frequency Selective Fading'); xlabel('Frequency(Hz)');
ylabel('Amplitude(V)');
%Fast and Flat Fading
fd = 10;%doppler frequency
hs = randi([0 1],1,length(ts)) +(1i*randi([0 1],1,length(ts))); [b,a] =
butter(12,((2*fd)/1000));
lpf = filter(b,a,hs); trans3 = lpf.*message;
magnitudeplot = abs(fft(trans3)); figure; plot(1:fs,magnitudeplot); title('Fast and Flat
Fading');
xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');
% Fast and Frequency Selective Fading
ht = randi([0 1],1,length(ts)) +1i*(randi([0 1],1,length(ts))); lpf1 = filter(b,a,ht);
trans4 = (lpf1.*message) +(lpf1.*delayed); magnitudeplot = abs(fft(trans4));
figure; plot(1:fs,magnitudeplot);
title('Fast and Frequency Selective Fading'); xlabel('Frequency(Hz)');
ylabel('Amplitude(V)');
```

<u>BLOCK DIAGRAM</u>:

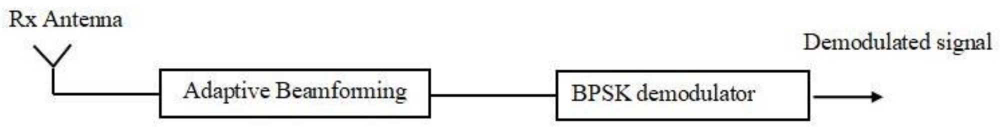
Transmitter:



Channel:



Receiver:



<u>ALGORITHM:</u>
BER VS SNR PERFORMANCE OF CHANNELS

1. Generate BPSK modulated wave.
2. Generate noise sequence that are gaussian distributed.
3. Multiply the modulated wave with the Rayleigh coefficient and add the noise.
4. Detect the received signal, compare it with the input signal to find the error.
5. Plot the BER vs SNR graph for Rayleigh channel.
6. Multiply the modulated wave with the rician coefficient and add the noise.
7. Detect the received signal, compare it with the input signal to find the error.
8. Plot the BER vs SNR graph for rician channel.

<u>RICIAN DISTRIBUTION:</u>

A Rician distribution is one way to model the paths scattered signal stake to a receiver. Specifically, this distribution models line-ofsight scatter — transmissions between two stations in view of each other that have an unobstructed path between them.Line-of-sight scatter includes FM radio waves,microwaves,MRI images in the presence of noise, and satellite transmissions. The distribution also models Rician fading, which is a way to show how signal cancellations affect radio propagation. The probability density function formula is: RAYLEIGH DISTRIBUTION Rayleigh distribution is a continuous probability distribution for positive-valued random variables. The data can be given by the mean value and a lower bound, or by a parameter and a lower bound. These are interconnected by a well-documented relationship given in the literature. For instance, if the mean μ=2 and the lower bound is γ=0.5, then θ=1.59577 and the standard deviation is σ=1.0454.

$$f(x \mid \nu, \sigma) = \frac{x}{\sigma^2} \exp\left(\frac{-(x^2 + \nu^2)}{2\sigma^2}\right) I_0\left(\frac{x\nu}{\sigma^2}\right),$$

<u>RAYLEIGH DISTRIBUTION</u>

Rayleigh distribution is a continuous probability distribution for positive-valued random variables. The data can be given by the mean value and a lower bound, or by a parameter and a lower bound. These are interconnected by a well-documented relationship given in the literature.For instance, if the mean μ=2 and the lower bound is γ=0.5, then θ=1.59577 and the standard deviation is σ=1.0454.

$$f(X) = \frac{X}{\beta^2} \exp\left[-\frac{1}{2}\left(\frac{X}{\beta}\right)^2\right]$$

INFERENCE:

- The amplitude remains the same in slow flat fading.
- The amplitude varies for slow and frequency selective fading.
- The amplitude remains the same but the signal is spread in fast and flat fading.
- The amplitude changes and the signal are spread over the frequencies in fast and frequencyselective fading.

RESULT:

Thus, we have simulated and plotted the spectra of four different types of fading and BER performance of Rayleigh & Rician channel were analyse using MATLAB and verified successfully

EXP NO: 8 **DATE:**	**NETWORK SECURITY PROTOCOLS RC4 ENCRYPTION**

AIM

To write a MATLAB program for rc4 encryption algorithm and check whether cipher that can be decrypted to get the plain text again.

SOFTWARE REQUIRED

MATLAB R2022b

THEORY

RC4 is a Stream Cipher and variable length key algorithm. This algorithm encrypts one byte at a time. A key input is pseudo random bit generator that produces a stream 8 bit number that is unpredictable without knowledge of input key, the output of generator is called key stream is combined one byte at a time with plain text stream cipher using Ex-Or operation.

PROCEDURE

1) Plain text and a variable length key are initialized.

2) Key scheduling algorithm: The entire of S are set equal to the values from 0-255 in ascending order, a temporary vector k is created. This step is called initialization step. If length of the key array is 255 bytes. Then key is assigned to k. For a key with length (k) bytes, the first k-len elements of k as copied from key and then key is repeated as many times as necessary to fill k. The code is written in left.

3) We use this k to produce the initial permutation of state vector s, starting with S[0] to S[255] and for each S[i], swap it with another byte in S according to a scheme dictated by K[i], but still S will contain values from 0 to 255.

4) Pseudo random generation algorithm: Once the vector S is initialized, the input ‘key’ will not be used. In this step, for each S[i] swap it with another bye in S according to a scheme dictated by the current configuration of S. After reaching S[255] the process continues starting from S[0] again.

5) By using key stream generated encrypt the message byte by byte doing the simple ex or operation.

6) At the receiver side, the cipher text received is again exored with same key stream to get the plaintext again. The effect of flat fading is seen as decrease in SNR. These flat fading channels areknown as amplitude varying channels or narrow band channels.

ALGORITHM

1) The entries of S are set equal to values from 5 to 255 in ascending order, a temporary vector T is created.

2) Once the vector S is initialized, the input key will not be used.

3) For each S[i], the algorithm sweeps it another byte in S. After reaching S[255], the process continues starting from S[0] again.

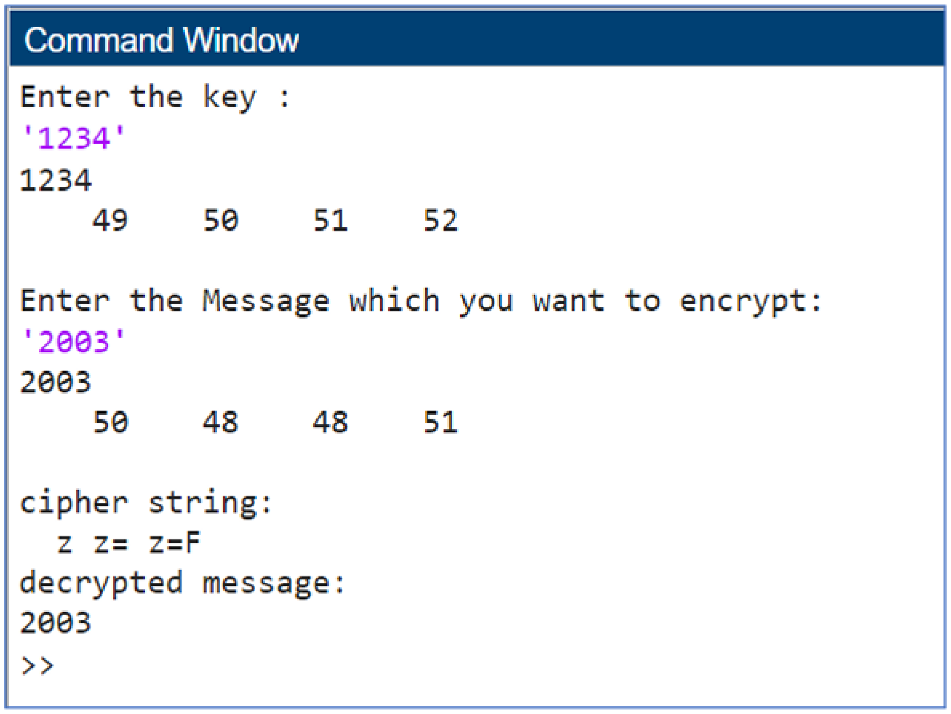
4) Encrypt using XOR to derive the cipher text.

CODE

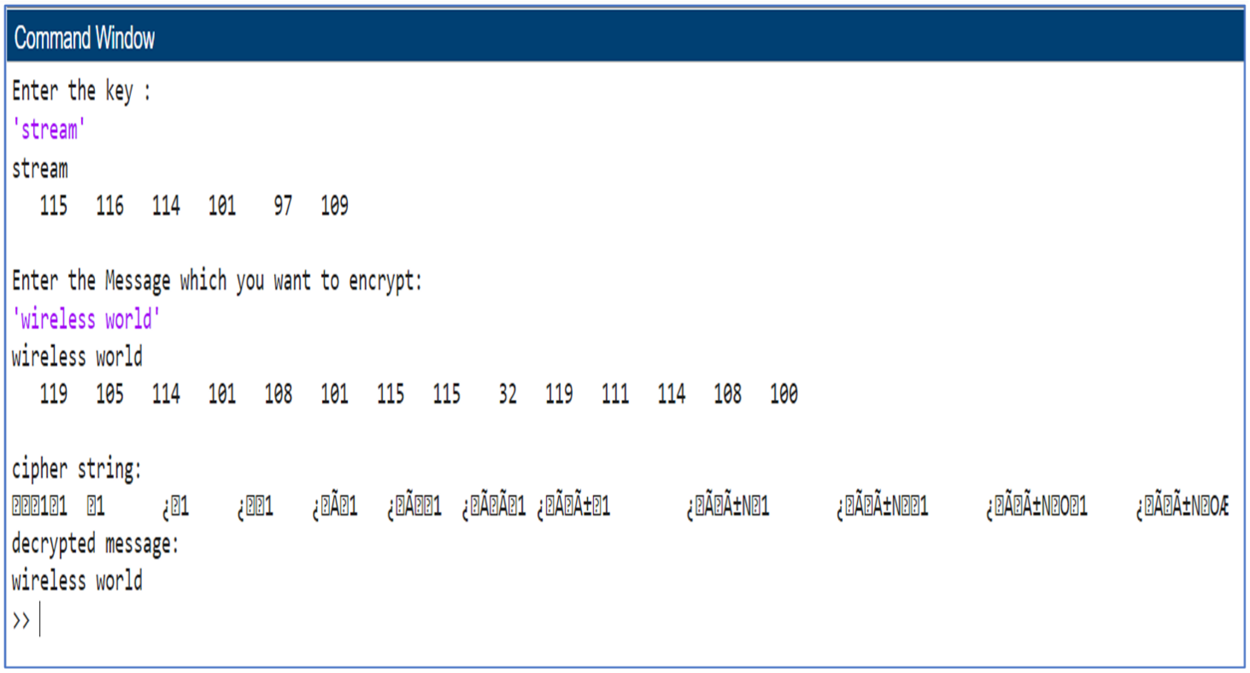
```
clc;clear;close all;

n = zeros(1,256);

str = input('Enter the key : ');

key = double(char(str));

disp(str);

disp(key);

k = zeros(1,256);

message = input('Enter the Message which you want to encrypt: ');

msg = double(char(message));

disp(message);

disp(msg);

keylen = length(key);

for i = 1:256

n(i) = i-1;
```

```
a = mod(i-1, keylen)+1;
k(i) = key(a);
end
j=0;
for i = 1:256
j = mod((i+n(i)+k(i)),256)+1;
n([i j]) = n([j i]);
end
ciphtext = [];
ciphtextstore = [];
keystream = [];
j = 0;
for i = 1:length(msg)
j = mod((j + n(i)), 256)+1;
n([i j]) = n([j i]);
a = mod((n(i)+n(j)),256)+i;
currentkey = n(i);
keystream = [keystream, currentkey];
ciphtext(i) = bitxor(msg(i), currentkey);
ciphtextstore = [ciphtextstore, ciphtext];
end
ciphertextstring = char(ciphtextstore);
disp("cipher string:")
disp(ciphertextstring);
decryptedmsg = [];
for p = 1:length(ciphtext)
currentmsg = bitxor(ciphtext(p), keystream(p));
```

OUTPUT



```
Enter the key :
'1234'
1234
    49    50    51    52

Enter the Message which you want to encrypt:
'2003'
2003
    50    48    48    51

cipher string:
  z z= z=F
decrypted message:
2003
>>
```



```
Enter the key :
'stream'
stream
   115   116   114   101    97   109

Enter the Message which you want to encrypt:
'wireless world'
wireless world
   119   105   114   101   108   101   115   115    32   119   111   114   108   100

cipher string:
□□□1□1  □1      ¿□1     ¿□□1    ¿□Ã□1    ¿□Ã□□1   ¿□Ã□Ã□1 ¿□Ã□Ã±□1         ¿□Ã□Ã±N□1         ¿□Ã□Ã±N□□1         ¿□Ã□Ã±N□O□1         ¿□Ã□Ã±N□OÆ
decrypted message:
wireless world
>> |
decryptedmsg = [decryptedmsg, currentmsg];
end
decryptedoutput = char(decryptedmsg);
disp('decrypted message:')
disp(decryptedoutput);
```

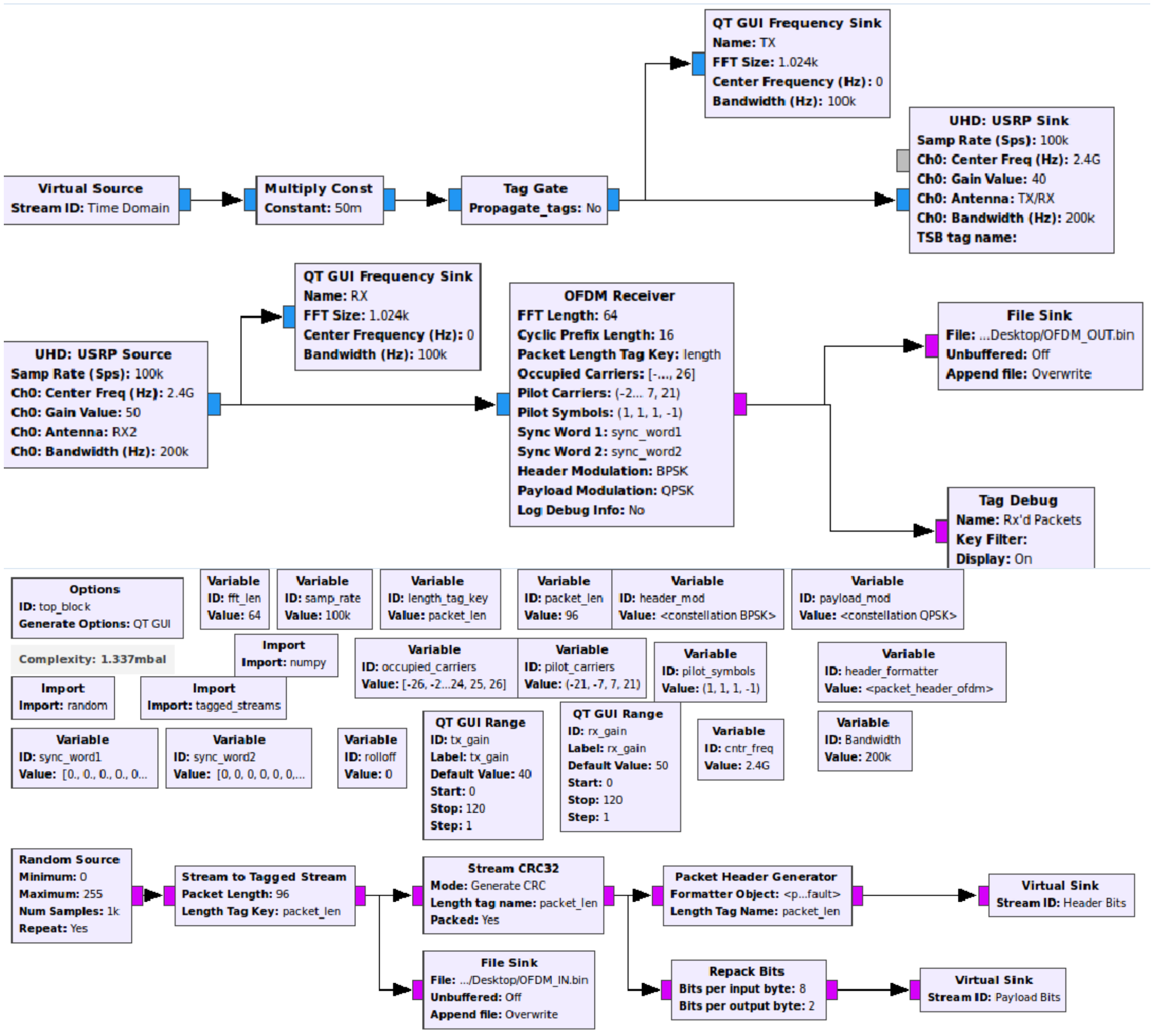
INFERENCE:

RC4 is a faster way of encrypting and verifying and is widely used to secure and sensitive data, particularly when it is being sent over an insecure network.

RESULT:

RC4 encryption algorithm is simulated and verified using MATLAB.

<u>OFDM Transmitter:</u>



QT GUI Frequency Sink
Name: TX
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 100k

UHD: USRP Sink
Samp Rate (Sps): 100k
Ch0: Center Freq (Hz): 2.4G
Ch0: Gain Value: 40
Ch0: Antenna: TX/RX
Ch0: Bandwidth (Hz): 200k
TSB tag name:

Virtual Source
Stream ID: Time Domain

Multiply Const
Constant: 50m

Tag Gate
Propagate_tags: No

QT GUI Frequency Sink
Name: RX
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 100k

OFDM Receiver
FFT Length: 64
Cyclic Prefix Length: 16
Packet Length Tag Key: length
Occupied Carriers: [-..., 26]
Pilot Carriers: (-2... 7, 21)
Pilot Symbols: (1, 1, 1, -1)
Sync Word 1: sync_word1
Sync Word 2: sync_word2
Header Modulation: BPSK
Payload Modulation: QPSK
Log Debug Info: No

File Sink
File: ...Desktop/OFDM_OUT.bin
Unbuffered: Off
Append file: Overwrite

UHD: USRP Source
Samp Rate (Sps): 100k
Ch0: Center Freq (Hz): 2.4G
Ch0: Gain Value: 50
Ch0: Antenna: RX2
Ch0: Bandwidth (Hz): 200k

Tag Debug
Name: Rx'd Packets
Key Filter:
Display: On

Options
ID: top_block
Generate Options: QT GUI

Variable
ID: fft_len
Value: 64

Variable
ID: samp_rate
Value: 100k

Variable
ID: length_tag_key
Value: packet_len

Variable
ID: packet_len
Value: 96

Variable
ID: header_mod
Value: <constellation BPSK>

Variable
ID: payload_mod
Value: <constellation QPSK>

Complexity: 1.337mbal

Import
Import: numpy

Variable
ID: occupied_carriers
Value: [-26, -2...24, 25, 26]

Variable
ID: pilot_carriers
Value: (-21, -7, 7, 21)

Variable
ID: pilot_symbols
Value: (1, 1, 1, -1)

Variable
ID: header_formatter
Value: <packet_header_ofdm>

Import
Import: random

Import
Import: tagged_streams

Variable
ID: sync_word1
Value: [0., 0., 0., 0., 0...

Variable
ID: sync_word2
Value: [0, 0, 0, 0, 0, 0,...

Variable
ID: rolloff
Value: 0

QT GUI Range
ID: tx_gain
Label: tx_gain
Default Value: 40
Start: 0
Stop: 120
Step: 1

QT GUI Range
ID: rx_gain
Label: rx_gain
Default Value: 50
Start: 0
Stop: 120
Step: 1

Variable
ID: cntr_freq
Value: 2.4G

Variable
ID: Bandwidth
Value: 200k

Random Source
Minimum: 0
Maximum: 255
Num Samples: 1k
Repeat: Yes

Stream to Tagged Stream
Packet Length: 96
Length Tag Key: packet_len

Stream CRC32
Mode: Generate CRC
Length tag name: packet_len
Packed: Yes

Packet Header Generator
Formatter Object: <p...fault>
Length Tag Name: packet_len

Virtual Sink
Stream ID: Header Bits

File Sink
File: .../Desktop/OFDM_IN.bin
Unbuffered: Off
Append file: Overwrite

Repack Bits
Bits per input byte: 8
Bits per output byte: 2

Virtual Sink
Stream ID: Payload Bits

EXP NO: 9 **DATE:**	**Implementation of OFDM using SDR Kit**

AIM

To implement OFDM using Universal Software Radio Peripheral (USRP)- Software Defined Radio(SDR) kit..

HARDWARE REQUIRED

USRP-SDR Kit

SOFTWARE REQUIRED

GNU Radio Software Suite

THEORY

a) Software Defined Radio:

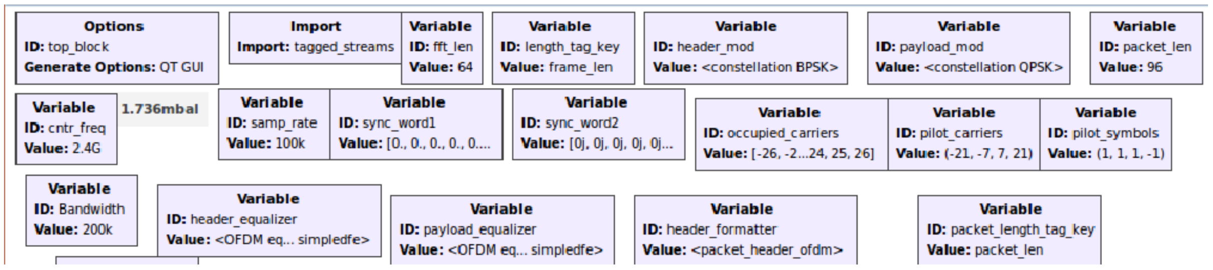
Software-defined radio (SDR) is a radio communication system where components that have been traditionally implemented in analog hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer. While the concept of SDR is not new, the rapidly evolving capabilities of digital electronics render practical many processes which were once only theoretically possible.

A basic SDR system may consist of a personal computer equipped with a sound card, or other analog-to-digital converter, preceded by some form of RF front end. Significant amounts of signal processing are handed over to the general-purpose processor, rather than being done in special-purpose hardware (electronic circuits). Such a design produces a radio which can receive and transmit widely different radio protocols (sometimes referred to as waveforms) based solely on the software used.

b) GNU Radio:

GNU Radio is a free software development toolkit that provides signal processing blocks to implement software-defined radios and signal-processing systems. It can be

OFDM Receiver:



Options
ID: top_block
Generate Options: QT GUI

Import
Import: tagged_streams

Variable
ID: fft_len
Value: 64

Variable
ID: length_tag_key
Value: frame_len

Variable
ID: header_mod
Value: <constellation BPSK>

Variable
ID: payload_mod
Value: <constellation QPSK>

Variable
ID: packet_len
Value: 96

Variable
ID: cntr_freq
Value: 2.4G

1.736mbal

Variable
ID: samp_rate
Value: 100k

Variable
ID: sync_word1
Value: [0., 0., 0., 0., 0....

Variable
ID: sync_word2
Value: [0j, 0j, 0j, 0j, 0j...

Variable
ID: occupied_carriers
Value: [-26, -2...24, 25, 26]

Variable
ID: pilot_carriers
Value: (-21, -7, 7, 21)

Variable
ID: pilot_symbols
Value: (1, 1, 1, -1)

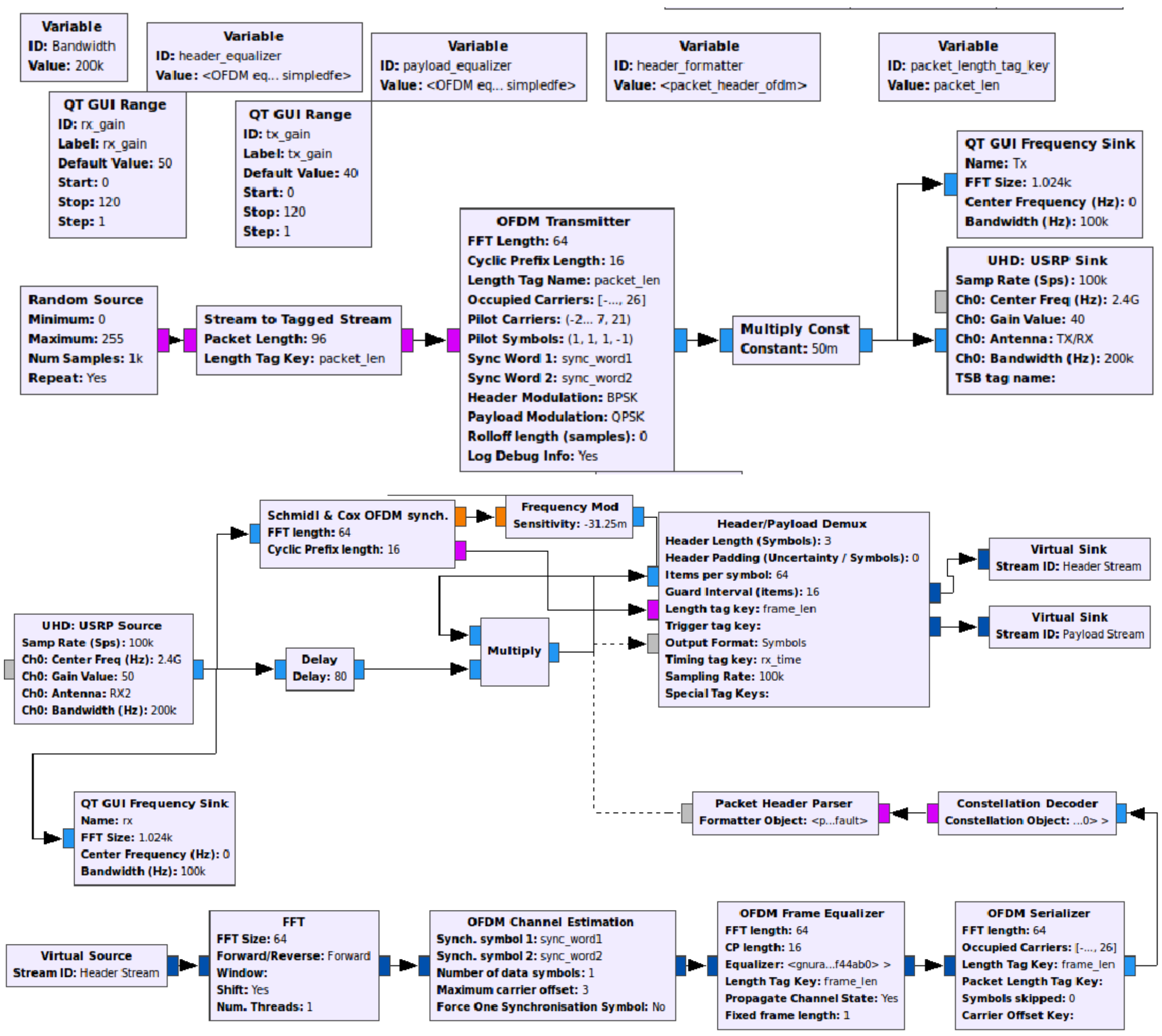
Variable
ID: Bandwidth
Value: 200k

Variable
ID: header_equalizer
Value: <OFDM eq... simpledfe>

Variable
ID: payload_equalizer
Value: <OFDM eq... simpledfe>

Variable
ID: header_formatter
Value: <packet_header_ofdm>

Variable
ID: packet_length_tag_key
Value: packet_len



Variable
ID: Bandwidth
Value: 200k

Variable
ID: header_equalizer
Value: <OFDM eq... simpledfe>

Variable
ID: payload_equalizer
Value: <OFDM eq... simpledfe>

Variable
ID: header_formatter
Value: <packet_header_ofdm>

Variable
ID: packet_length_tag_key
Value: packet_len

QT GUI Range
ID: rx_gain
Label: rx_gain
Default Value: 50
Start: 0
Stop: 120
Step: 1

QT GUI Range
ID: tx_gain
Label: tx_gain
Default Value: 40
Start: 0
Stop: 120
Step: 1

QT GUI Frequency Sink
Name: Tx
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 100k

OFDM Transmitter
FFT Length: 64
Cyclic Prefix Length: 16
Length Tag Name: packet_len
Occupied Carriers: [-..., 26]
Pilot Carriers: (-2... 7, 21)
Pilot Symbols: (1, 1, 1, -1)
Sync Word 1: sync_word1
Sync Word 2: sync_word2
Header Modulation: BPSK
Payload Modulation: QPSK
Rolloff length (samples): 0
Log Debug Info: Yes

Random Source
Minimum: 0
Maximum: 255
Num Samples: 1k
Repeat: Yes

Stream to Tagged Stream
Packet Length: 96
Length Tag Key: packet_len

Multiply Const
Constant: 50m

UHD: USRP Sink
Samp Rate (Sps): 100k
Ch0: Center Freq (Hz): 2.4G
Ch0: Gain Value: 40
Ch0: Antenna: TX/RX
Ch0: Bandwidth (Hz): 200k
TSB tag name:

Schmidl & Cox OFDM synch.
FFT length: 64
Cyclic Prefix length: 16

Frequency Mod
Sensitivity: -31.25m

Header/Payload Demux
Header Length (Symbols): 3
Header Padding (Uncertainty / Symbols): 0
Items per symbol: 64
Guard Interval (items): 16
Length tag key: frame_len
Trigger tag key:
Output Format: Symbols
Timing tag key: rx_time
Sampling Rate: 100k
Special Tag Keys:

Virtual Sink
Stream ID: Header Stream

Virtual Sink
Stream ID: Payload Stream

UHD: USRP Source
Samp Rate (Sps): 100k
Ch0: Center Freq (Hz): 2.4G
Ch0: Gain Value: 50
Ch0: Antenna: RX2
Ch0: Bandwidth (Hz): 200k

Delay
Delay: 80

Multiply

QT GUI Frequency Sink
Name: rx
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 100k

Packet Header Parser
Formatter Object: <p...fault>

Constellation Decoder
Constellation Object: ...0> >

Virtual Source
Stream ID: Header Stream

FFT
FFT Size: 64
Forward/Reverse: Forward
Window:
Shift: Yes
Num. Threads: 1

OFDM Channel Estimation
Synch. symbol 1: sync_word1
Synch. symbol 2: sync_word2
Number of data symbols: 1
Maximum carrier offset: 3
Force One Synchronisation Symbol: No

OFDM Frame Equalizer
FFT length: 64
CP length: 16
Equalizer: <gnura...f44ab0> >
Length Tag Key: frame_len
Propagate Channel State: Yes
Fixed frame length: 1

OFDM Serializer
FFT length: 64
Occupied Carriers: [-..., 26]
Length Tag Key: frame_len
Packet Length Tag Key:
Symbols skipped: 0
Carrier Offset Key:

used with external RF hardware to create software-defined radios, or without hardware in a simulation-like environment. It is widely used in hobbyist, academic, and commercial environments to support both wireless communications research and real-world radio systems. The GNU Radio software provides the framework and tools to build and run software radio or just general signal-processing applications. The GNU Radio applications themselves are generally known as "flowgraphs", which are a series of signal processing blocks connected together, thus describing a data flow.

c) <u>Software Defined Radio</u>:

Software-defined radio (SDR) is a radio communication system where components that have been traditionally implemented in analog hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer. While the concept of SDR is not new, the rapidly evolving capabilities of digital electronics render practical many processes which were once only theoretically possible.

A basic SDR system may consist of a personal computer equipped with a sound card, or other analog-to-digital converter, preceded by some form of RF front end. Significant amounts of signal processing are handed over to the general-purpose processor, rather than being done in special-purpose hardware (electronic circuits). Such a design produces a radio which can receive and transmit widely different radio protocols (sometimes referred to as waveforms) based solely on the software used.

d) <u>GNU Radio</u>:

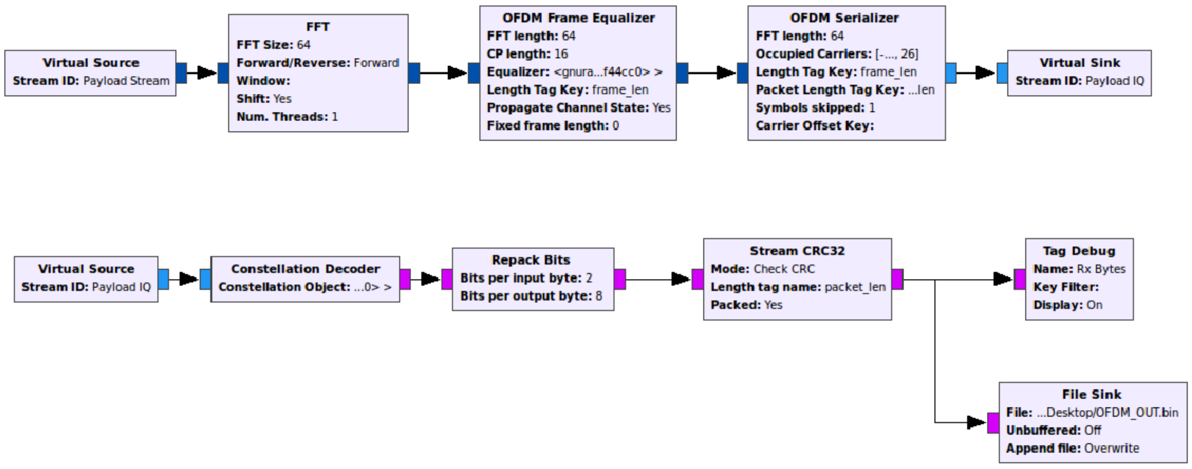
GNU Radio is a free software development toolkit that provides signal processing blocks to implement software-defined radios and signal-processing systems. It can be used with external RF hardware to create software-defined radios, or without hardware in a simulation

-like environment. It is widely used in hobbyist, academic, and commercial environments to support both wireless communications research and real-world radio systems. The GNU Radio software provides the framework and tools to build and run software radio or just general signal-processing applications. The GNU Radio applications themselves are generally known as "flowgraphs", which are a series of signal processing blocks connected together, thus describing a data flow.

Flowgraphs:

<u>Transmission</u>:

The transmission flow graph is shown in transmitter Figures. We convert a stream of digital data into packets and append ’headers/tags’ to these packets for identification. This is followed by converting the data into a constellation scheme as shown in Figure . Further processing to convert it into OFDM signal to transmit through the channel is done by the flow graph shown in Figure . A group of channel-coded bits are gathered together


Virtual Source
Stream ID: Payload Stream

FFT
FFT Size: 64
Forward/Reverse: Forward
Window:
Shift: Yes
Num. Threads: 1

OFDM Frame Equalizer
FFT length: 64
CP length: 16
Equalizer: <gnura...f44cc0> >
Length Tag Key: frame_len
Propagate Channel State: Yes
Fixed frame length: 0

OFDM Serializer
FFT length: 64
Occupied Carriers: [-..., 26]
Length Tag Key: frame_len
Packet Length Tag Key: ...len
Symbols skipped: 1
Carrier Offset Key:

Virtual Sink
Stream ID: Payload IQ

Virtual Source
Stream ID: Payload IQ

Constellation Decoder
Constellation Object: ...0> >

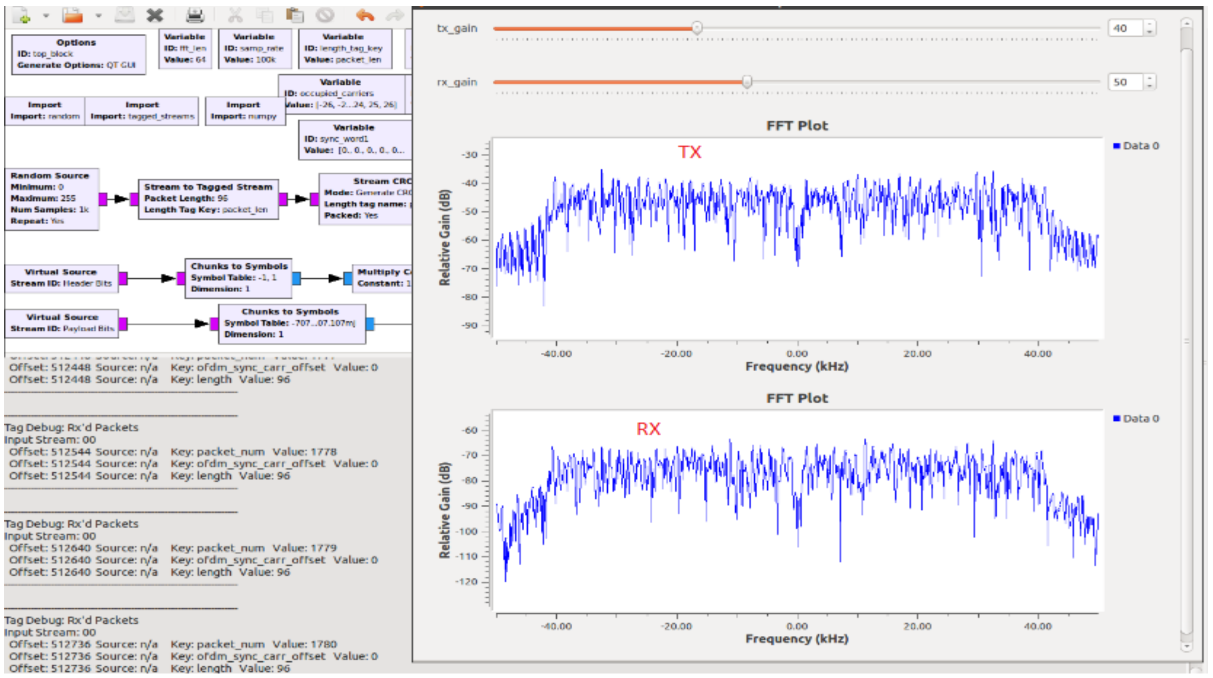
Repack Bits
Bits per input byte: 2
Bits per output byte: 8

Stream CRC32
Mode: Check CRC
Length tag name: packet_len
Packed: Yes

Tag Debug
Name: Rx Bytes
Key Filter:
Display: On

File Sink
File: ...Desktop/OFDM_OUT.bin
Unbuffered: Off
Append file: Overwrite

Plot outputs:



Options
ID: top_block
Generate Options: QT GUI

Variable
ID: fft_len
Value: 64

Variable
ID: samp_rate
Value: 100k

Variable
ID: length_tag_key
Value: packet_len

Variable
ID: occupied_carriers
Value: [-26, -2...24, 25, 26]

Import
Import: random

Import
Import: tagged_streams

Import
Import: numpy

Variable
ID: sync_word1
Value: [0., 0., 0., 0., 0...

Random Source
Minimum: 0
Maximum: 255
Num Samples: 1k
Repeat: Yes

Stream to Tagged Stream
Packet Length: 96
Length Tag Key: packet_len

Stream CRC
Mode: Generate CRC
Length tag name:
Packed: Yes

Virtual Source
Stream ID: Header Bits

Chunks to Symbols
Symbol Table: -1, 1
Dimension: 1

Multiply C
Constant: 1

Virtual Source
Stream ID: Payload Bits

Chunks to Symbols
Symbol Table: -707...07.107mj
Dimension: 1

```
 Offset: 512448 Source: n/a    Key: ofdm_sync_carr_offset  Value: 0
 Offset: 512448 Source: n/a    Key: length  Value: 96
--------------------------------------------------------------

--------------------------------------------------------------
Tag Debug: Rx'd Packets
Input Stream: 00
 Offset: 512544 Source: n/a    Key: packet_num   Value: 1778
 Offset: 512544 Source: n/a    Key: ofdm_sync_carr_offset  Value: 0
 Offset: 512544 Source: n/a    Key: length  Value: 96
--------------------------------------------------------------

--------------------------------------------------------------
Tag Debug: Rx'd Packets
Input Stream: 00
 Offset: 512640 Source: n/a    Key: packet_num   Value: 1779
 Offset: 512640 Source: n/a    Key: ofdm_sync_carr_offset  Value: 0
 Offset: 512640 Source: n/a    Key: length  Value: 96
--------------------------------------------------------------

--------------------------------------------------------------
Tag Debug: Rx'd Packets
Input Stream: 00
 Offset: 512736 Source: n/a    Key: packet_num   Value: 1780
 Offset: 512736 Source: n/a    Key: ofdm_sync_carr_offset  Value: 0
 Offset: 512736 Source: n/a    Key: length  Value: 96
```

(1 for BPSK, 2 for QPSK, 4 for 16-QAM, etc.) and mapped to corresponding constellation points. At this stage, the data are represented as complex numbers and they are in serial. Known pilot symbols mapped with known mapping schemes can be inserted at this moment.

A serial to parallel conversion is then applied and IFFT operation is performed on the obtained parallel complex data. The output data of this operation are grouped together again, as per the number of required transmission subcarriers. Cyclic prefix is inserted in every block of data according to the system specification and the data are multiplexed in a serial fashion. At this stage, the data are OFDM modulated and ready to be transmitted. A USRP is used to transform the time-domain digital data, up-convert it to transmission frequency and to send it across a wireless channel.

Reception:

Receiver blocks describe the flowgraphs for receiving the transmitted OFDM signal, converting it back to the constellation scheme, and extracting the payload bits out of the received packets. After the transmission of OFDM signal from the transmitter antenna, the signals go through all the anomaly and hostility of wireless channel. After receiving the signal, the receiver USRP down-converts the signal and converts it to digital domain. While down-converting the received signal, carrier frequency synchronization is performed. After this conversion, symbol timing synchronization is achieved.

An FFT block is then used to demodulate the OFDM signal. After that, channel estimation is performed using the demodulated pilots. Using the estimations, the complex data are obtained which are demapped according to the transmission constellation diagram.

RESULT

Thus we have implemented OFDM using USRP- SDR Kit and plotted output.

EXP NO: 10 **DATE:**	**SPECTRUM SENSING FOR COGNITIVE RADIO NETWORK**

AIM

To study the spectrum for cognitive radio network using USRP-SDR Kit.

HARDWARE REQUIRED

USRP B200 kit

SOFTWARE REQUIRED

GNU Radio Software Suite

THEORY

a) **Cognitive Radio Network:**
A cognitive radio (CR) is a radio that can be programmed and configured dynamically to use the best channels in its vicinity to avoid user interference and congestion. Such a radio automatically detects available channels, then accordingly changes its transmission or reception parameters to allow more concurrent wireless communications in a given band at one location. This process is a form of dynamic spectrum management.

b) **Spectrum Sensing:**
Spectrum sensing enables a cognitive radio to have information about its environment and spectrum availability. The most widely used spectrum sensing methods are energy detection and matched filter detection. It is a critical issue of cognitive radio technology because of the shadowing, fading, and time-varying natures of wireless channels.

Primary Users:

Primary Users are the incumbent users who have higher priority or legacy rights over specific portions of the radio frequency spectrum. They are

Sensed Output:

SPECTRUM SENSING – SENSED OUTPUT

```
-08-11  13:17:23.868116   92700000.0   92678125.0   3.47082781698    -105.111235933
-08-11  13:17:23.868146   92700000.0   92681250.0   4.32710233399    - 105.111235933
-08-11  13:17:23.868177   92700000.0   92684375.0   5.23300214568    - 105.111235933
-08-11  13:17:23.868207   92700000.0   92687500.0   5.21400197619    - 105.111235933
-08-11  13:17:23.868238   92700000.0   92690625.0   4.88849130871    -103.738065933
-08-11  13:17:23.868268   92700000.0   92693750.0   6.47467957612    -103.738065933
-08-11  13:17:23.868299   92700000.0   92696875.0   8.534982987       -103.738065933
-08-11  13:17:23.868330   92700000.0   92700000.0   10.1640703974    -102.334165933
-08-11  13:17:23.868361   92700000.0   92703125.0   10.076343398     -102.334165933
-08-11  13:17:23.868392   92700000.0   92706250.0   7.33869500533   -102.334165933
-08-11  13:17:23.868470   92700000.0   92709375.0   4.78562950634    -103.738065933
-08-11  13:17:23.868501   92700000.0   92712500.0   5.80400205289    -103.738065933
-08-11  13:17:23.868532   92700000.0   92715625.0   4.6245779801     -103.738065933
-08-11  13:17:23.868562   92700000.0   92718750.0   4.94535769506   -103.738065933
-08-11  13:17:23.868593   92700000.0   92721875.0   5.15780699035   - 105.111235933
-08-11  13:17:23.868623   92700000.0   92725000.0   5.95893395367   - 105.111235933
```

GNU Radio with USRP B200

typically licensed to use specific frequency bands for communication services such as cellular networks, TV broadcasting, or emergency services. PUs is the original occupants of the spectrum and has the first claim to its usage. Spectrum sensing aims to detect the presence or absence of PUs' signals.

Secondary Users:

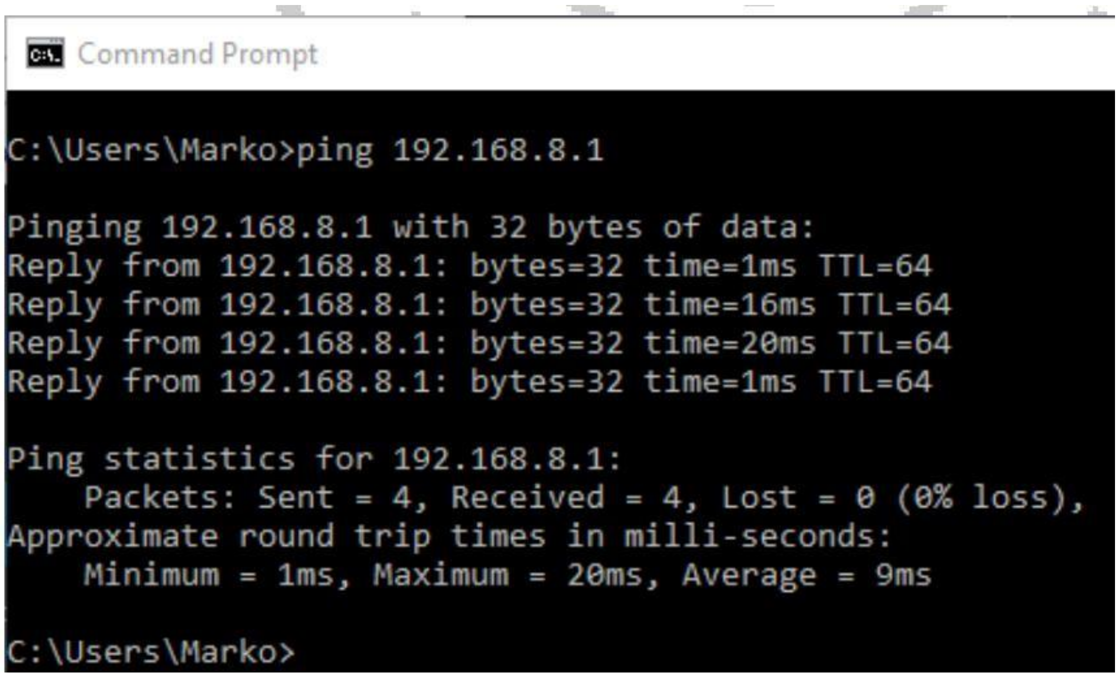
Secondary Users, on the other hand, have lower priority compared to PUs. They are equipped with cognitive radio capabilities, allowing them to sense the surrounding spectrum. SUs opportunistically exploit the spectrum that is unused by PUs. When SUs detect an available frequency band, they can transmit without causing harmful interference to PUs. SUs must vacate the spectrum promptly if PUs reoccupies it.

In summary, PUs holds the primary rights, while SUs operate opportunistically in the available spectrum, ensuring efficient utilization and coexistence in cognitive radio-based systems.

Result:

Thus the spectrum is sensed for cognitive radio network and studied using GNU Radio Software Suite and USRP B200 kit.

Sample Output for Ping:



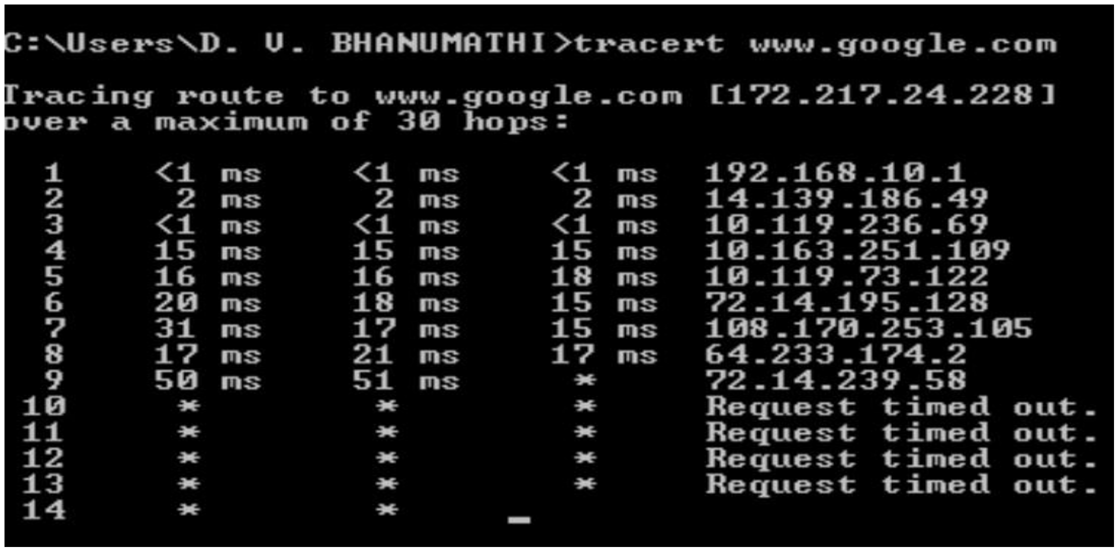
Command Prompt

```
C:\Users\Marko>ping 192.168.8.1

Pinging 192.168.8.1 with 32 bytes of data:
Reply from 192.168.8.1: bytes=32 time=1ms TTL=64
Reply from 192.168.8.1: bytes=32 time=16ms TTL=64
Reply from 192.168.8.1: bytes=32 time=20ms TTL=64
Reply from 192.168.8.1: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.8.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 20ms, Average = 9ms

C:\Users\Marko>
```

Four packets are sending to the destination and the destination responds back with the same four packets. 32 bytes of data are sent out and 32 bytes of data are got back in 9 milliseconds average.

Sample Output for Tracert :

tracert www.google.com



```
C:\Users\D. V. BHANUMATHI>tracert www.google.com

Tracing route to www.google.com [172.217.24.228]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.10.1
  2     2 ms     2 ms     2 ms  14.139.186.49
  3    <1 ms    <1 ms    <1 ms  10.119.236.69
  4    15 ms    15 ms    15 ms  10.163.251.109
  5    16 ms    16 ms    18 ms  10.119.73.122
  6    20 ms    18 ms    15 ms  72.14.195.128
  7    31 ms    17 ms    15 ms  108.170.253.105
  8    17 ms    21 ms    17 ms  64.233.174.2
  9    50 ms    51 ms     *     72.14.239.58
 10     *        *        *     Request timed out.
 11     *        *        *     Request timed out.
 12     *        *        *     Request timed out.
 13     *        *        *     Request timed out.
 14     *        *      _
```

EXP NO: 11 **DATE:**	**IMPLEMENTATION OF PING,TRACERT AND NSLOOKUP**

AIM

To study and implement the basic networking commands such as Ping, tracertand nslookup.

COMPONENT REQUIRED

PC

THEORY

Network troubleshooting with ping

The “ping” command is a subset of the ICMP (Internet Control Message Protocol), which allows us to send a signal to another device, and if that device is active, it will send a response back to the sender. It verifies IP-level connectivity to another TCP/IP computer by sending ICMP Echo Request messages. And if the device that is pinged is active or online, an echo response will be obtained.

For example, if the local computer has Internet connectivity issues, ping is used to test the router.

Tracert / traceroute

This diagnostic tool determines the path taken to a destination by sending ICMP EchoRequest messages with varying Time to Live (TTL) values to the destination. Each router along the path is required to decrement the TTL in an IP packet by at least 1 before forwarding it.

Syntax

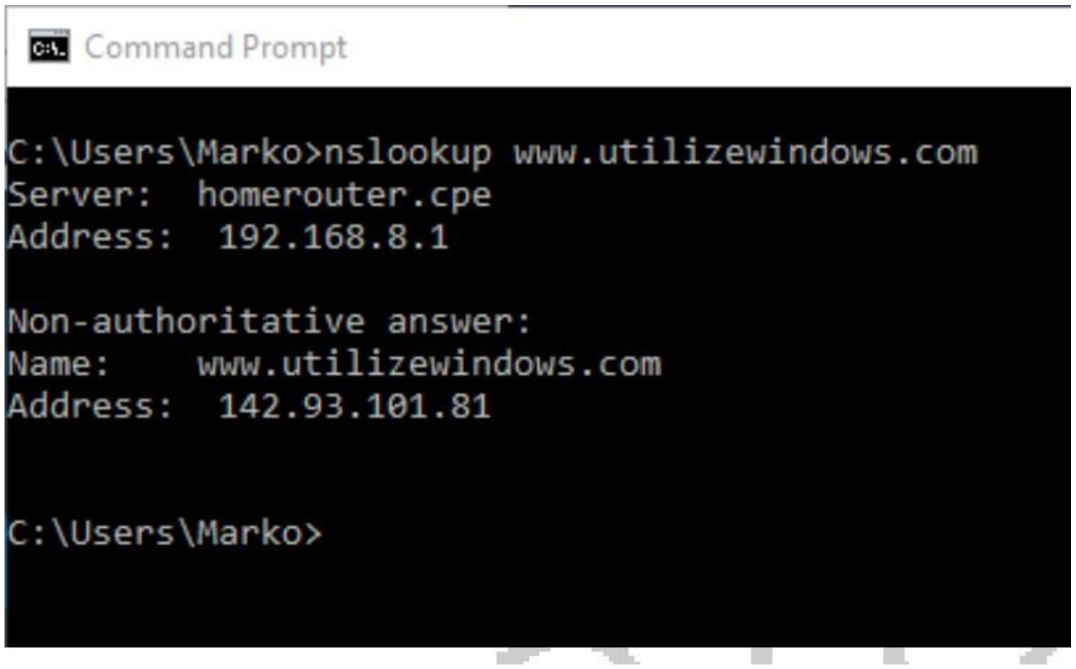
tracert [-d] [-h MaximumHops] [-j HostList] [-w Timeout] [TargetName]

Parameters

-d	Prevents tracert from attempting to resolve the IP addresses ofintermediate routers to their names. This can speed up the display of tracert results.

<u>Sample Output for nslookup :</u>

nslookup www.utilizewindows.com



```
C:\Users\Marko>nslookup www.utilizewindows.com
Server:  homerouter.cpe
Address:  192.168.8.1

Non-authoritative answer:
Name:    www.utilizewindows.com
Address:  142.93.101.81


C:\Users\Marko>
```

-h	Maximum Hops Specifies the maximum number of hops in the path to search for the target (destination). The default is30 hops.
-j	Host List Specifies that Echo Request messages use the Loose Source Route option in the IP header with the set of intermediate destinations specified in HostList. With loose source routing, successive intermediate destinations can be separated by one or multiple routers. The maximum number of addresses or names in the host list is 9. The Host List is a series of IP addresses (in dotted decimal notation) separated by spaces.
-w	Timeout Specifies the amount of time in milliseconds to wait for the ICMP Time Exceeded or Echo Reply message corresponding to a given Echo Request message to be received. If not received within the time-out, an asterisk (*) is displayed. The default time-out is 4000 (4 seconds)

nslookup:

The nslookup command will fetch the DNS records for a given domain name or an IP address. Remember the IP addresses and domain names are stored in DNS servers,so the nslookup command is used to query the DNS records to gather information. Toknow the IP address of www.utilizewindows.com, type in nslookup and type in www.utilizewindows.com.

RESULT :

Thus the IP address of a system is studied and implemented